December 2018

A Smart Products Lifecycle Management (sPLM) Framework - Modeling for Conceptualization, Interoperability, and Modularity

Yunpeng Li
Syracuse University

ABSTRACT

Autonomy and intelligence have been built into many of today's mechatronic products, taking advantage of low-cost sensors and advanced data analytics technologies. Design of product intelligence (enabled by analytics capabilities) is no longer a trivial or additional option for the product development. The objective of this research is aimed at addressing the challenges raised by the new data-driven design paradigm for smart products development, in which the product itself and the smartness require to be carefully co-constructed.

A smart product can be seen as specific compositions and configurations of its physical components to form the body, its analytics models to implement the intelligence, evolving along its lifecycle stages. Based on this view, the contribution of this research is to expand the "Product Lifecycle Management (PLM)" concept traditionally for physical products to data-based products. As a result, a Smart Products Lifecycle Management (sPLM) framework is conceptualized based on a high-dimensional Smart Product Hypercube (sPH) representation and decomposition.

First, the sPLM addresses the interoperability issues by developing a Smart Component data model to uniformly represent and compose physical component models created by engineers and analytics models created by data scientists. Second, the sPLM implements an NPD^3 process model that incorporates formal data analytics process into the new product development (NPD) process model, in order to support the transdisciplinary information flows and team interactions between engineers and data scientists. Third, the sPLM addresses the issues related to product definition, modular design, product configuration, and lifecycle management of analytics models, by adapting the theoretical frameworks and methods for traditional product design and development.

An sPLM proof-of-concept platform had been implemented for validation of the concepts and methodologies developed throughout the research work. The sPLM platform provides a shared data repository to manage the product-, process-, and configuration-related knowledge for smart products development. It also provides a collaborative environment to facilitate transdisciplinary collaboration between product engineers and data scientists.

A SMART PRODUCTS LIFECYCLE MANAGEMENT (SPLM) FRAMEWORK – MODELING FOR CONCEPTUALIZATION, INTEROPERABILITY, AND MODULARITY

by

Yunpeng Li

B.S., Beihang University, 2000
M.S., Syracuse University, 2014

Dissertation
Submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Mechanical and Aerospace Engineering

Syracuse University
December 2018

ACKNOWLEDGEMENTS

My first thank should be to my advisor, Prof. Utpal Roy. This research work would not be accomplished without his years of guidance, assistance, and support. I would never forget the time when he sat with me in the conference room, for many hours, to discuss and improve my first publication. I appreciate Prof. Roy to provide all the opportunities for conference presentations and industrial exposure. I thank Dr. Bicheng Zhu, Dr. Heng Zhang, Mr. Omer Yaman, Mr. Kai Sun, and Mr. Hang Yin in Prof. Roy's lab for all the research discussions and collaborations.

I would like to thank Professor Riyad Aboutaha, Prof. Young Moon, Prof. Jianshun Zhang, Prof. John Dannenhoffer, and Prof. Jeffrey Saltz for serving as my committee members and providing valuable questions, suggestions, and most importantly, encouragement for improving this research work. I specially thank Prof. Jeffrey Saltz and Prof. Jeffrey Stanton in iSchool for their guidance in the area of data science and data analytics. I thank Prof. Svetoslava Todorova and Prof. Michael Roppo for their guidance during the UAS project.

This work would not have been possible without the financial support from the National Institute of Standards and Technology (NIST) and the sponsors from New York State. I would like to thank Dr. Sudarsan Rachuri and Mrs. Tina Lee for inspiring and supporting the work. I also appreciate all the colleagues in NIST for research collaborations in the smart, sustainable manufacturing area. Specifically, I thank Dr. Seung-Jun Shin for providing example data and many feedbacks in the early stage of my research. I also thank the anonymous reviewers in NIST who helped sharpen the research ideas and improve my writing skills.

I would like to thank my family. I thank my beloved wife, Yanhua Jiang, for her dedication to support the family and take care of the two beloved kids, Kangbo Li and Daniel Li, for so many years. I am so proud of the journey of my PhD study had inspired Kangbo to pursue his college life in engineering. I thank my parents, Yinzhu Li and Xiuchang Bai, my parents-in-law, Renguo Jiang and Hengyu Huang, as well as my younger brother, Yungang Li, for their remote support. I am proud of being a member of the family.

Last but not least, I would thank my friends, Sonny Zhan's family, Biao Chen's family, Hong Wan' family, Dr. Richard Chiang's family, Pastor Mark Harrison, and many others with whom my family and I had so many pleasant memories in Syracuse, New York, during the past years.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

AHRS	Attitude and Heading Reference System
AI	Artificial Intelligence
ALM	Application Lifecycle Management
AML	Aras Markup Language
ANN	Artificial Neural Network
AP	Application Protocol
APF	Artificial Potential Field
API	Application Programming Interface
BO	Business Object
BOL	Beginning of Life
BOM	Bill of Materials
BPEL	Business Process Execution Language
BPMN	Business Process Model and Notation
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CE	Concurrent Engineering
CL_2M	Closed-Loop Lifecycle Management
CLIPS	C Language Integrated Production System
CNC	Computer Numerical Control
CP	Canonical Polyadic
CPM	Core Product Model
CPS	Cyber-Physical Systems
CRISP-DM	Cross-Industry Standard Process for Data Mining
CSP	Constraint Satisfaction Problem
CZML	Cesium Markup Language
DA	Data Analytics
DAIM	Design-Analysis Integration Model
DCSP	Dynamic Constraint Satisfaction Problem
DM	Data Mining
DMG	Data Mining Group
DMM	Domain Mapping Matrix
DMN	Decision Model and Notation
DPD	Data Products Development
DSM	Design Structure Matrix
ECN	Engineering Change Notice
EKF	Extended Kalman Filter

EOL	End-of-life
ERP	Enterprise Resource Planning
ETL	Extract, Transform, and Load
FAA	Federal Aviation Administration
FPV	First Person View
GCS	Ground Control Station
GD&T	Geometric Dimensioning and Tolerancing
GPR	Gaussian Process Regression
GUID	Globally Unique Identifier
HIL	Hardware-in-the-loop
HVAC	Heating, Ventilation, and Air Conditioning
ICT	Information and Communication Technology
ICUAS	International Conference on Unmanned Aircraft Systems
IEC	International Electrotechnical Commission
IGES	Initial Graphics Exchange Specification
IIoT	Industrial Internet of Things
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IoT	Internet of Things
IP	Intellectual Property
IPD	Integrated Product Development
ISO	International Organization for Standardization
IT	Information Technology
JSON	JavaScript Object Notation
KDDM	Knowledge Discovery and Data Mining
LCA	Life Cycle Assessment
MBD	Model-Based Definition
MES	Manufacturing Execution Systems
ML	Machine Learning
MLM	Model Lifecycle Management
MOL	Middle-of-Life
NATO	North Atlantic Treaty Organization
NIST	National Institute of Standards and Technology
NPD	New Product Development
NPD^3	New Product Development with Data-Driven Features
NTF	New-To-Firm
OAM	Open Assembly Model
OMG	Object Management Group
OWL	Ontology Web Language

p2u	product-to-user
p2p	product-to-product
PCA	Principal Component Analysis
PCB	Printed Circuit Board
PDP	Product Development Process
PEID	Product Embedded Information Device
PFA	Portable Format for Analytics
PFEM	Product Family Evolution Model
PITL	PLM-in-the-loop
PLCS	Product Life Cycle Support
PLE	Product Line Engineering
PLM	Product Lifecycle Management
PM	Project Management
PMI	Product and Manufacturing Information
PMML	Predictive Model Markup Language
PR	Perspective Reduction
PSS	Product-Service System
RAMI4.0	Reference Architecture Model for Industry 4.0
RF	Radio Frequency
RFID	Radio Frequency Identification
RRT	Rapidly-exploring Random Tree
SBP	Sampling-Based Planner
SDMDA	Sensor, Data, Model, Decision, and Actuator
SITL	Software-in-the-loop
sComponent	Smart Component
sPCM	Configuration Model for Smart Products
sPDM	Product Data Model for Smart Products
sPH	Smart Products Hypercube
sPLM	Smart Products Lifecycle Management
sPPM	Process Model for Smart Products Development
STEP	STandard for the Exchange of Product model data
UAS	Unmanned Aircraft System
UAV	Unmanned Aircraft Vehicle
UTM	UAS Traffic Management
XML	Extensible Markup Language

Chapter 1

Introduction

The transformative power arising out of the fusion of information and communication technologies (ICT) including sensor networks, big data analytics and cloud computing, has changed the way a product is developed, manufactured, serviced and managed throughout the product's lifecycle. Products are getting smarter with the capabilities to perform reasoning based on known knowledge and to learn new knowledge from past experience (Li et al., 2015b). With sensors and complicated algorithms, a household thermostat can, for example, autonomously establish a mathematical model that captures a building's inside thermal dynamics, without prior knowledge about the building characteristics such as it's size, layout, leakiness, and HVAC (heating, ventilation, and air conditioning) system (Nest Labs, 2012). Equally interesting, an unmanned aircraft vehicle can establish an occupancy map of its environment and can sense and avoid obstacles. As these two examples demonstrate, data and the capabilities to process data into knowledge and decisions have become critical components of the product itself and of the process to develop/operate the product.

The need for smart products to monitor, control and provide adaptation capabilities sets them apart from traditional products. The coordination needed across product design, cloud operation, service improvement, and customer engagement is continuous and never ends, even after the sale (Porter and Heppelmann, 2015). Often times, the use of sensors within smart products provides the data needed for intelligence. Data analytics provides the tools and technologies needed to increase the intelligence of the device (Li et al., 2015b). New data-centered product design and development paradigms have been emerging to inform the traditional processes and for the development of data-driven products. In the *data-informed* design paradigm, data can be utilized to reveal patterns and trends to drive innovation,

measure product performance, and incrementally improve the product experience (Pavliscak, 2015). As a result, a data-centered design approach can improve the development and operations of the product. On the other hand, data can be the "material" being processed by machine learning algorithms to produce *data-driven* products (e.g. predictive or prescriptive analytics models), which in turn, can generate more data (Patil, 2012; Dhar, 2013). In this case, data is used to create data-driven machine learning features within a smart product.

The two paradigms have been collectively used in the development of modern smart mechatronics systems. For instance, automotive companies are employing these data-centered design techniques for the development of a car's autopilot capability as well as to improve the car's reliability (Geiger and Sarakakis, 2016). Consequently, the new discipline, *Data Science*, and the new experts, *Data Scientists*, are emerging and need to be incorporated into the product development team (Porter and Heppelmann, 2015).

In the following sections, we first review the product evolution trend and discuss the product capability classifications based on different perspectives. We then present a harmonized view of smart products definitions and their characteristics. Furthermore, we discuss the roles of data analytics and how they contribute to smart products' intelligence. We use a real-world consumer product, the *Nest Self-Learning Thermostat*, to illustrate the concepts throughout the discussion, in order to establish the background contexts for the research. Finally, the observations from the smart thermostat motivate us to further study the implications and challenges raised by the data-centered product design paradigm.

1.1 Product Evolution

The information and communication technologies have evolved fast in the past half century. The first two waves of ICT-driven transformations arose during the 1960s-1970s and the 1980s-1990s,

accompanied with the emergence of computers and Internet. These information technologies (IT) had enabled the automation, coordination and integration of individual activities in the manufacturing value chain that is across computer-aided design, manufacturing resource planning, logistics, and after-sales services. They also drove the third industrial revolution that was commenced in the 1970s[1]. Industry is now undergoing the third-wave IT-driven transformation (Gens, 2013; Morris et al., 2014), in which embedded sensors, processors, software, and connectivity in products, coupled with product clouds in which product data is stored and analyzed and many software-driven or data-driven applications are running (Porter and Heppelmann, 2014; 2015). This is driving dramatic improvements in product functionality and performance. In Germany, this new-wave technology is called "Industry 4.0" (MacDougall, 2013); and in the United States, it is called "Advanced Manufacturing" or "Smart Manufacturing" (PCAST, 2014; Hehenberger et al., 2016).

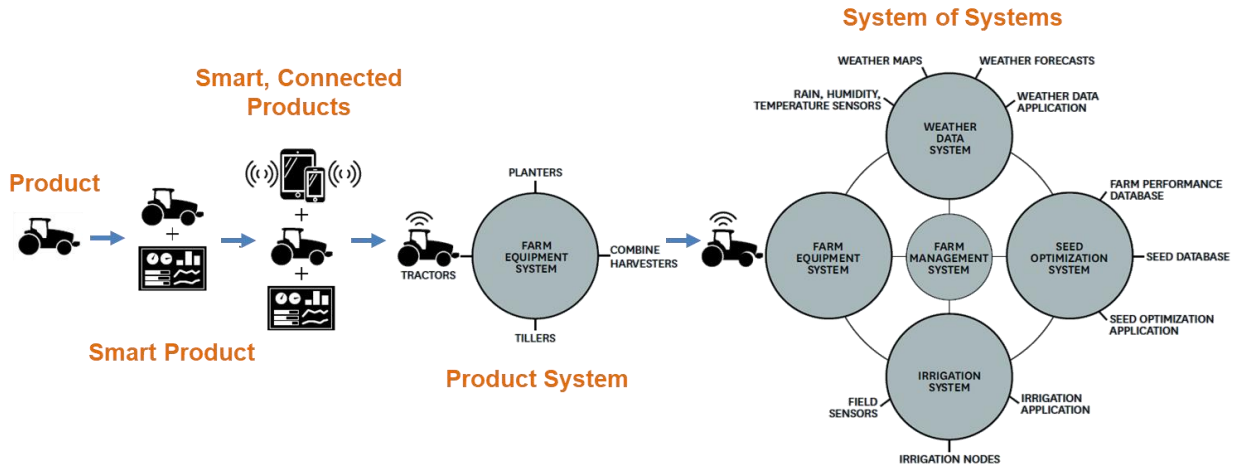


Figure 1.1 Product to smart, connected product to system of systems (adapted from Porter and Heppelmann, 2014)

Low-cost electronic components and almost ubiquitous wireless connectivity make it both technically and economically feasible to embed computing and networking functionality into almost any

[1] Note: The first industrial evolution commenced at the end of the 18th century with the introduction of mechanical production equipment, and the second industrial evolution commenced at the turn of the 20th century with the advent of electrically powered machinery for mass production.

object – from industrial equipment and vehicles to wearables and home appliances (EIU, 2015). Smart products development has not only to address the technological challenges including ever-growing smartness and connectivity capabilities of products, it is also changing the relationship that companies have with their customers, thus is transforming their organizational structure and business models. Furthermore, the increasing capabilities of smart, connected products not only reshape competition within industries but expand industry boundaries. This occurs as the basis of competition shifts from discrete products, to product systems consisting of closely related products, to system of systems that link an array of product systems together. A tractor company, for example, may have to reposition itself to collaborate or compete in a broader farm automation industry (Figure 1.1).

1.1.1 Technological Perspective: *Smartness and Connectivity*

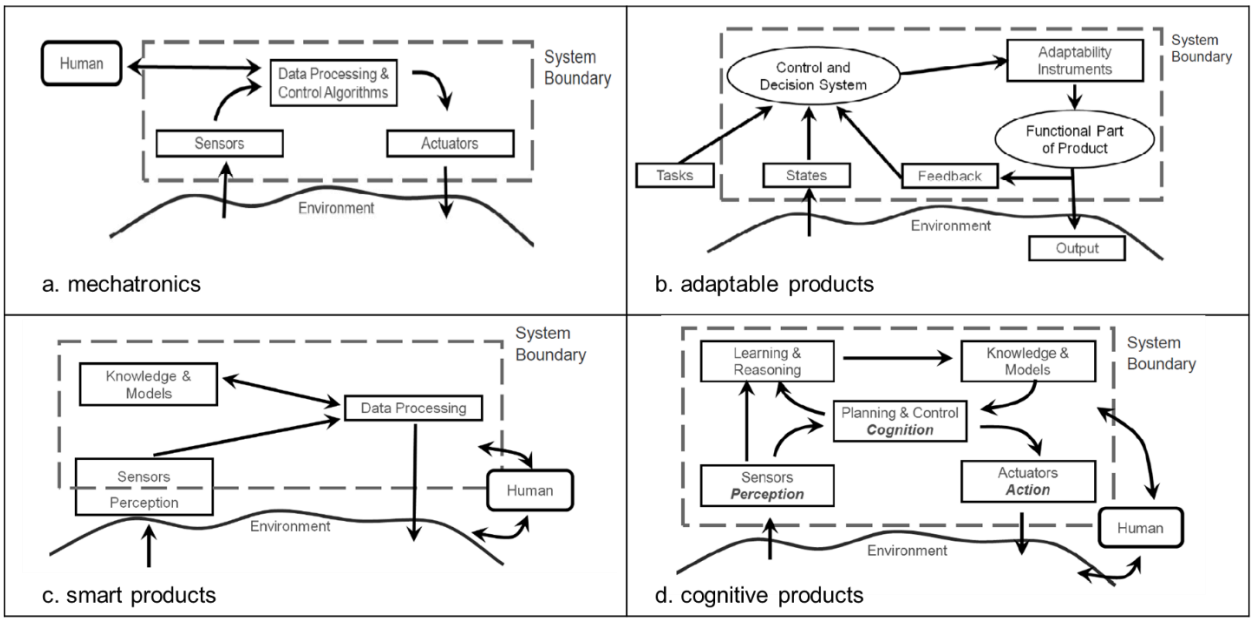


Figure 1.2 Evolution of product smartness: Mechatronics to adaptable products to smart products to cognitive products (adapted from Beetz et al., 2007; Metzler and Shea, 2010)

Traditionally, products had been classified by their components according to the field of engineering they belong to, e.g. mechanical-, electronical-, and software engineering. Nowadays, products are commonly differentiated by their capabilities and not solely by their components (Metzler

and Shea, 2010). According to their capabilities, products can be classified as mechatronic, adaptable, smart, or even autonomous systems due to the increasing "intelligence" that stems from embedded software, hardware, and advanced data processing capabilities.

Specifically, as shown in Figure 1.2, *Mechatronics* are multidisciplinary complex systems combining mechanical, electrical and software components. A key characteristic of mechatronic systems is the functional integration of sensors, actuators and data processing; therefore, it is a critical enabler for automation. *Adaptable Products* are mechatronic systems that accommodate predictable changes the designers have foreseen therefore they can derive appropriate action plans. They execute the pre-programmed algorithms in the control systems and carry out the action plans using a combination of adaptable instruments. The adaptability can take place during different phases of the product lifecycle: design time, runtime, and lifetime. The adaptation of a product's lifetime is usually achieved by prolonging the service life in its normal operational mode and by adapting it to new operational modes (as being seen from the product-service systems discussed in next section).

Smart products have further abilities that differentiate them from adaptable products. They can perceive the environment using sensors inside or outside the product as well as receiving environment data from external devices or data sources. The combination of knowledge and perception in terms of sensory data enables smart products to know their current state, i.e. situational awareness. The degrees of their situational awareness are determined by the quantity and quality of sensors, and often times, data/information fusion techniques are employed to increase the accuracy of state estimate (Castanedo, 2013). The situational awareness in turn allows these products to interact, network and communicate with their environment including humans, robots, or other products as needed. However, the early smart products only possess weak artificial intelligence (AI) that is focused on narrow tasks. More recently, *Cognitive products* are emerging with increasing autonomous capabilities. A cognitive product also

collects data of their environment and their own status through sensors, and then processes and interprets this data in order to generate an appropriate response through actuators. But the goal of a cognitive product is to achieve similar levels of flexibility, adaptivity, and robustness as found in the most cognitive system, *humans*. The cognition of a product is that all data is processed and interpreted according to the perceived situation and not according to a rigid control algorithm. For instance, an autonomous unmanned aircraft vehicle can establish an occupancy map of its environment, and is able to sense and avoid obstacles by dynamically replanning its flight trajectory, if needed.

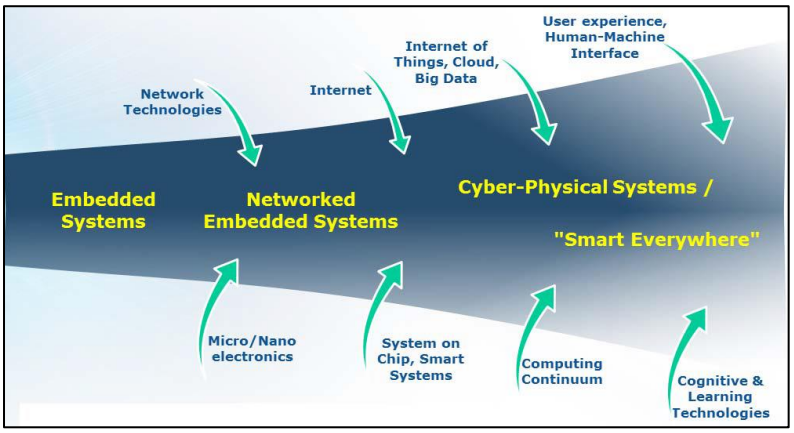


Figure 1.3 Evolution of product connectivity: Embedded systems to networked embedded systems to cyber-physical systems (adapted from Geisberger and Broy, 2014)

Products can also be classified by the scale of their connectivity (Figure 1.3). *Cyber-Physical Systems (CPS)* are defined as intelligent mechatronic products/systems capable of communicating and interacting with other CPS by using different communication channels (the Internet or local network). CPS is an enabling technology that brings the virtual and physical worlds together to create a networked world in which intelligent objects communicate and interact with each other. It also tightly integrates the ability of computing, communication, and control on the basis of information acquisition of Internet of Things (Seshia et al., 2017).

The *Internet of Things (IoT)* is the network of physical devices, vehicles, and other items embedded

with electronics, software, sensors, actuators, and network connectivity which enable these objects to collect and exchange data[2]. The *Industrial Internet of Things (IIoT)*, also known as the *Industrial Internet*, brings together brilliant machines, advanced analytics, and people at work. It is the network of a multitude of devices constructing a system of systems that can monitor, collect, exchange, analyze, and deliver valuable new insights. These insights can then help derive smarter, faster business decisions for industrial companies[3].

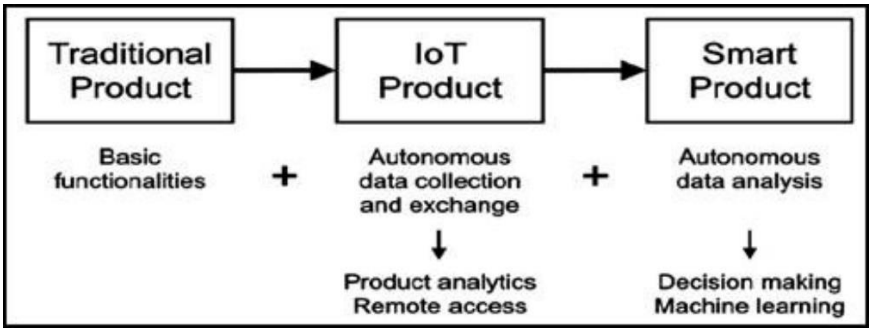


Figure 1.4 Product to IoT to Smart Product (Decker and Stummer, 2017)

The boundaries between these product capability (smartness and connectivity) classifications are vague. They are linked to one another and the transition from one schema to another is fuzzy and depends on individual contexts. For example, as shown in Figure 1.4, IoT products can be seen as an interim stage in the development of smart products (Decker and Stummer, 2017). In this view, IoT products are capable of collecting and sharing data via the Internet and providing information to users; but smart products are able to analyze usage data in order to learn and adapt to customer preferences over time. More specifically, IoT products have two elementary functionalities: *product analytics* and *remote access*. Product analytics relies on the autonomous collection of usage data to provide the manufacturer with insights into the actual product use. Remote access enables remotely operating the

[2] Internet of Things, https://en.wikipedia.org/wiki/Internet_of_things
[3] Everything you need to know about the Industrial Internet of Things, https://www.ge.com/digital/blog/everything-you-need-know-about-industrial-internet-things

IoT product with changing the parameters or adjustment of product attributes, activating/deactivating product functions, and controlling data flowing through the IoT product. Smart products have two more elementary functionalities: *learning* and *decision making*. The decisions made by a smart product can be used to provide users with recommendations rather than just information as in the case of an IoT product.

1.1.2 Business Perspective: *Product and Services*

The monitoring, control, and adaptation capabilities of smart products are raising the "Servitization" trends of products. Servitization is recognized as the process of creating value by adding services to products. The first use of this term in the context of manufacturing operations was by Vandermerwe and Rada (1988). They defined servitization as "*the increased offering of fuller market packages or 'bundles' of customer focused combinations of goods, services, support, self-service and knowledge in order to add value to core product offerings*." In contrast, "Productization" is the evolution of the services component to include a product or a new service component marketed as a product.

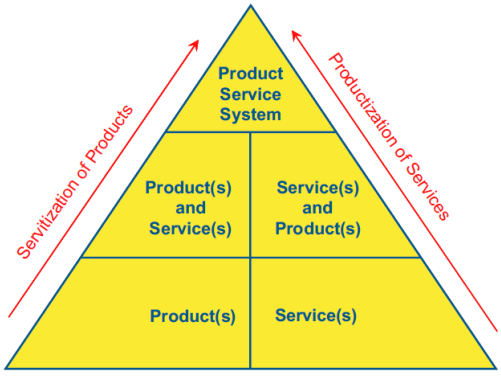


Figure 1.5 Evolution of the PSS concept (Baines et al., 2007)

The convergence of servitization and productization results in a *Product-Service System (PSS)* that considers a product and a service as a single offering (see Figure 1.5). The early definition of PSS came from Goedkoop et al. (1999):

"*A product-service system is a system of products, services, networks of players and supporting infrastructure that continuously strives to be competitive, satisfy customer needs and have lower environmental impact than traditional business models.*"

A PSS can also be seen as a special case of servitization, which values asset performance or utilization rather than ownership (Baines et al., 2007). There are three types of PSS solutions based on different combination-levels of products and services (Tukker, 2004): (1) *Product-oriented* PSS is a supply of products that comes with extra services. Examples of this category include maintenance, repair, reuse and recycling; (2) *Use-oriented* PSS is selling the use or availability of a product that is not owned by the customer (e.g., leasing, sharing, pooling); and (3) *Result-oriented* PSS is selling a result or capability instead of a product (e.g., selling laundered clothes instead of a washing machine).

A well-known PSS example is the TotalCare package offered by Rolls-Royce (R-R) to airlines. R-R delivers the "power-by-the-hour" gas turbine technology rather than transferring ownership of the gas turbine engine to the airlines (Smith, 2013). R-R maintains direct access to the assets thus they can collect data on product performance and use (Figure 1.6). Such data can then enable the improvement of performance parameters (e.g., maintenance schedules) to improve engine efficiency, asset utilization, and reduce total cost and the environmental impact. In this case, the services developed based upon data not only add smartness to the engines; the services themselves are also products sold as commodities.

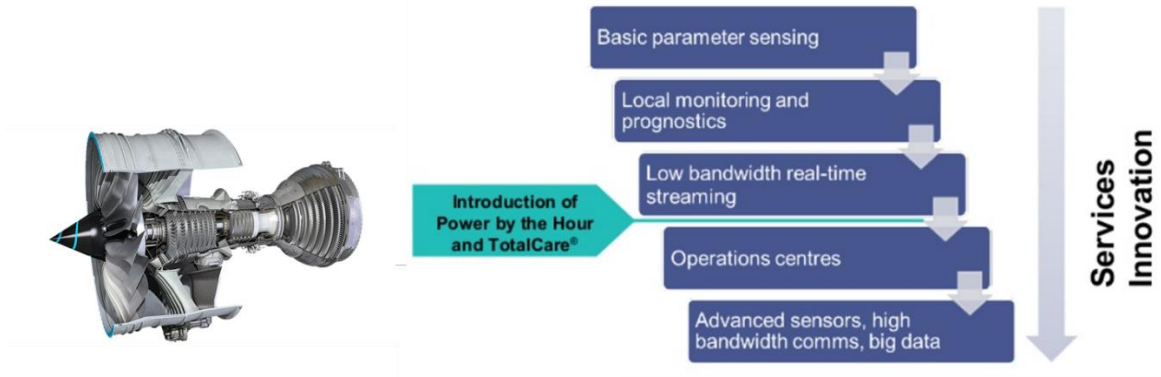


Figure 1.6 Rolls-Royce TotalCare (Paul, 2013)

That is, the increasing levels of product analytics and product smartness are enabling more service-

oriented capabilities of the products because it is transforming the relationship that companies have with the customers. The recent Industry 4.0 is speeding up the integration of technical processes and business processes, thus it speeding up the trend of product-services convergence. The Industry 4.0 is also blurring the boundary between products and productions. The technological evolution from embedded systems to CPS is enabling more "decentralized" production through extensively using the Internet (MacDougall, 2013). Eventually, the Industry 4.0 requires the digital mapping and virtualization of the real world, the data-centered paradigm in turn paves the path from developing networked smart products to building the Internet of smart services, see Figure 1.7.

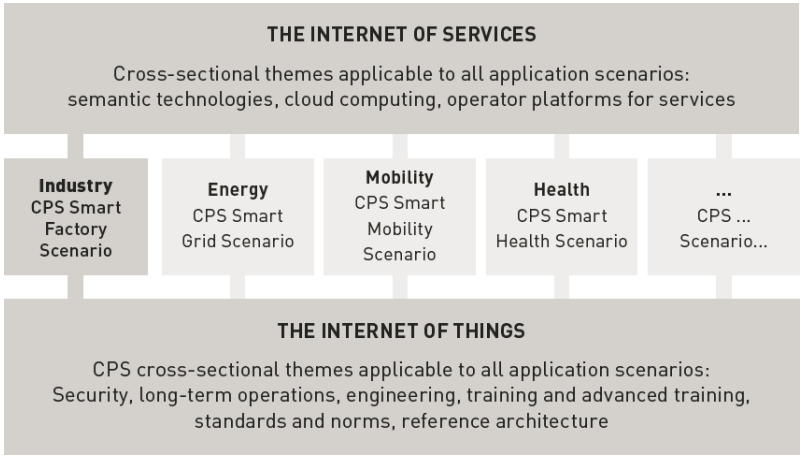


Figure 1.7 The landscape of Industry 4.0 (MacDougall, 2013)

1.2 Defining Smart Products

To date, there has been no universal way to define the capability of a smart product. For instance, the "Smart Products" term is often interchangeably used with another term "Intelligent Products" (Gutierrez et al. 2013). As discussed earlier, the mechatronics, adaptive, smart/intelligent, or cognitive products can be classified based on different perspectives: smartness, connectivity, and business models. They are linked to one another and the transition between schemas depends on individual contexts.

To understand the characteristics of smart products, below we study a real-world smart product, the

Nest Thermostat (see Appendix A for details). We use this self-learning thermostat example to illustrate a harmonized view of the definitions for smart products.

1.2.1 A Smart Product: *Nest Self-learning Thermostat*

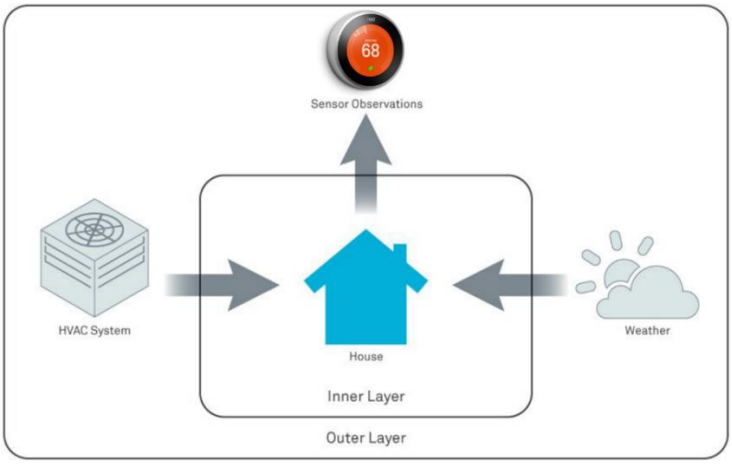


Figure 1.8 The Nest Thermostat thermal model (Nest, 2015)

The primary customer need for a residential thermostat is to achieve greater energy savings while maintaining the user's comfort. However, literature had reported that many residential thermostats failed to achieve energy savings even though they could be automated via programming because users tend not to use the feature (Peffer et al., 2011). The Nest Thermostat was the first self-learning thermostat that implemented a smart feature called *Auto-Schedule* (Lohr, 2011) to fill the gap. It employs a sophisticated machine learning algorithm that can automatically learn the user's living pattern and generate an energy-efficient yet comfortable control strategy to the linked HVAC system (Figure 1.8). This auto-schedule feature together with its supporting smart features (*Auto-Away* detection, *Time-to-Temperature* estimation, etc.), as well as the underlying data and computing infrastructure, form a smart ecosystem named *Nest Sense*. The Nest sense is further augmented by the *Nest Cloud* to make it capable of incorporating third-party services like local weather forecasting. It is noted that each Nest Thermostat doesn't have any prior knowledge about the building's characteristics such as the size, layout, leakiness,

and its HVAC system. It has to build a model that captures the thermal dynamics inside the house based on sensed data.

According to Nest Labs (2012-2014), the first-generation auto-schedule feature was developed via simulation. The simulation model consisted of physics-based models (including heat transfer model, air infiltration model and weather model, heating/cooling equipment model) and data-driven analytics models (auto-away, auto-schedule, time-to-temperature) to capture the dynamics of the environment in which the thermostat had been installed. Three years later after the first release, an *Enhanced Auto-Schedule* feature was released. This upgrade was a result of utilizing the accumulated actual usage data collected from many houses across different climate regions, thus more accurately captured thermal dynamics and users' behaviors (Nest Labs, 2014). This enhanced auto-schedule feature had been upgraded on all three generations of Nest Thermostats in service ever since 2011, without introducing new hardware components.

The Nest Thermostat shows many intelligent capabilities including *self-learning*, *self-planning*, *continuous improvement*, *autonomy*, and *swarming as a group of thermostats*. While it is simple regarding the physical structure, it leverages the complexity of machine learning algorithms and data analytics to understand the physical world and the user's behavior. Each installed instance also networks with other thermosets and external services to form a complex, smart, connected system.

1.2.2 Definition of Smart Products

As early as 2005, Allmendinger and Lombreglia investigated the notion of smartness in a product from the business perspective. They regarded smartness as *the product's capability to predict errors and faults thus "removing unpleasant surprises from the users' lives"*. This earlier definition viewed smart products as augmented everyday objects with sensors, actuators, displays, and computational elements,

embedding in a smart, ubiquitous computing environment (Sabou et al., 2009). On the other side, the smartness definitions from the product perspective are often related to the products' autonomy. For instance, the National Institution of Standards and Technology (NIST) defines the autonomy of an unmanned aircraft system (UAS) as

> *"the UAS's own abilities of sensing, perceiving, analyzing, communicating, planning, decision-making, and acting/executing, to achieve its goals as assigned by its human operator(s) through a designed human-robot interface or by another system that the UAS communicates with."* (Huang, 2008)

To strengthens this product perspective, Mühlhäuser (2008) identified two motivating goals for building smart products. On one hand, there is an increasing need for *Simplicity* throughout the entire lifecycle of the product, as its functionalities become ever more complex. On the other hand, the increasing number, sophistication and diversity of product components require a considerable level of *Openness* on the product's side. The simplicity and the openness need be achieved with improved product-to-user (p2u) and product-to-product (p2p) interactions. This implies the smartness needs be "carefully designed within the product" during the product development process. The actual product and the corresponding smartness need to be "co-constructed" at the conceptualization and system design stages; later in the lifecycle, knowledge held by the smart product has to include both "constructed" and "accumulated" parts (Mühlhäuser, 2008). A smart product was thus defined as:

> *"An entity (tangible object, software, or service) designed and made for self-organized embedding into different (smart) environments in the course of its lifecycle, providing improved simplicity and openness through improved product-to-user and product-to-product interactions by means of context-awareness, semantic self-description, proactive behavior, multimodal natural interfaces, AI planning, and machine learning."* (Mühlhäuser, 2008)

In order to provide an industry-applicable, lifecycle-spanning methodology to support the construction of smart products, SmartProducts Consortium (Sabou et al., 2009) consolidates Mühlhäuser's definition and other early definitions (Kärkkäinen et al., 2003; McFarlane et al., 2003;

Ventä, 2007) to depict several fundamental characteristics of a smart product. A smart product should (1) be identifiable; (2) be able to retain or store data about itself; (3) continuously monitor its status and environment; (4) react and adapt to environmental and operational conditions; (5) maintain optimal performance; and (6) actively communicate with the user, environment, and/or other products and systems. Based on these characteristics, a smart product definition is extended and harmonized as:

"A Smart Product is an autonomous object which is designed for self-organized embedding into different environments in the course of its lifecycle and which allows for a natural product-to-human interaction. Smart products are able to proactively approach the user by using sensing, input and output capabilities of the environment thus being self-, situational-, and context-aware. The related knowledge and functionality can be shared by and distributed among multiple smart products and emerges over time." (Sabou et al., 2009)

1.2.3 Characteristics of Smart Products

The SmartProducts Consortium's definition focuses on products' autonomous capabilities and comprehensively captures the main characteristics of smart products. That is, the product can perceive, learn, communicate, adapt, and act with the environment throughout the product's lifecycle. The capability to incorporate emerging knowledge over time differentiates smart products from other types of products. Table 1.1 describes each characteristic in details and illustrates them using the Nest Thermostat example.

Table 1.1 Characteristics of smart products

Characteristics	***Description***	***Nest Thermostat Example***
Autonomy	Smart products need to be able to operate on their own without relying on a central infrastructure and central control.	Each installed Nest Thermostat can work alone to detect the environmental conditions and control the HVAC system.
Situational and contextual awareness	Smart products are able to sense physical, virtual information and to infer higher level events (termed as "situation") from the raw data.	The Nest Thermostat can predict the user's behavior from its sensor data and the data from the user's other devices connected to the network, for example, a smart phone.
Self-organized embedding in smart product environment	A smart product is able to embed itself into an existing smart environment and to automatically build a smart product environment.	A smart thermostat can embed itself into the building's HVAC system, capture the environmental dynamics of the building, then build a home model accordingly.
Proactively approach the user	The situation information is used to determine when and how the smart product should proactively approach the user to provide	The *Energy Service* resided in the *Nest Cloud* can notify users the energy rush hours that happen on extreme weather days

	information or assist performing tasks.	and tune the temperature on each individual Nest Thermostat, if needed.
Support the user throughout the whole lifecycle	The particular lifecycle stage of a product has a major influence on its behavior.	The ability of the Nest Thermostat to sense the user context during the use phase provides its usage history during the recycling phase.
Multimodal interaction	Smart products should be able to make use of the different input and output capabilities in their smart environment supporting the usage of various modalities for natural interaction.	The Nest Thermostat can show information on its own screen or on the screen of the user's smart phone.
Support procedural knowledge	Smart products need to support procedural knowledge, including how the user needs to be involved in the different steps and how implicit interaction can be integrated in the procedure. The procedures are not limited to one single smart product, the procedures can also be dynamically composed of procedures provided by several smart products.	The Nest Thermostat has predefined logics to guide the user to set up the device and remind the user if there is an exceptional event. The thermostat also has predefined programs incorporating physics-based knowledge, e.g., heat transfer model, air infiltration model and weather model, as well as heating/cooling equipment models.
Emerging knowledge	Smart products learn new knowledge from observing the user, incorporating user feedback and exploring other external knowledge sources. They are able to gather a more accurate user model and to learn new procedures.	Once a Nest Thermostat is deployed in service, it starts to learn and adapt. The Auto-Schedule feature generates a unique schedule for each individual user who is using the thermostat.
Distributed storage of knowledge	Smart products can outsource their knowledge to other smart products in the environment. The distributed storage also enables a new smart product can be initialized with knowledge of the old product once that product is decommissioned.	The Nest Cloud provide such a distributed platform for all installed thermostat instances.

1.2.4 Intelligence of Smart Products

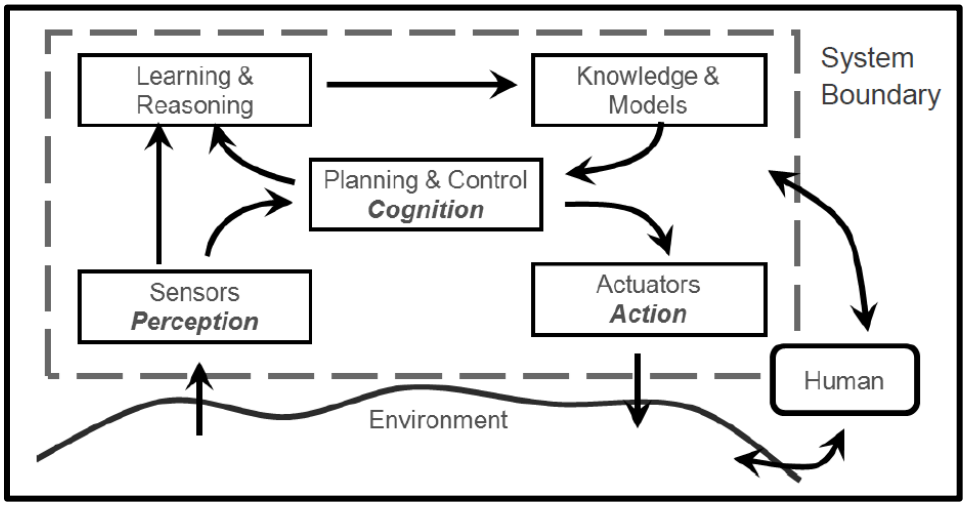


Figure 1.9 Key components of cognition (Metzler and Shea, 2010)

Similar as a human, a smart product needs to incorporate necessary cognitive capabilities to achieve

the characteristics defined previously. As shown in Figure 1.9, the fundamental components of a cognitive system include: (1) *Perception*: the acquisition of information about the environment and the body of an actor; (2) *Action*: the process of generating behavior to change the world and to achieve some objectives of the acting entity; (3) *Knowledge*: conceived to consist of both declarative and procedural knowledge; (4) *Learning*: the process of acquiring information, and, respectively, the reorganization of information that results in new knowledge (emerging knowledge); (5) *Reasoning*: the cognitive process by which an individual or system may infer a conclusion from an assortment of evidence, or from statements of principles; and (6) *Planning*: the process of generating representations of future behavior, prior to the use of such plans.

Each component requires necessary functions regarding data and information processing, modeling, and decision making. Table 1.2 shows the observed cognitive capabilities from the Nest Thermostat.

Table 1.2 Capabilities of the Nest Thermostat

Capability	***Data, Information, and Cognition***	***NEST Thermostat Example***
Perception	location	●
	user recognition	●
	object recognition	
Data Processing & Control	data processing	●
	control	●
Knowledge	environment model	●
	user model	●
	model of itself	●
Learning	locations	●
	point of time	●
	behavior model	●
	use of resources	●
	user models	●
Reasoning	task fulfillment	●
	alternative actions	●
Planning	route	
	schedule	●
	use of resources	●
Action	autonomous movement	
	object manipulation	●
Communication & Interaction	user adaption	●
	teach user	●
	give feedback	●
	swarm	●

Meyer et al. (2008) classified the intelligence of smart products into three dimensions: *the level of*

intelligence, *the location of intelligence*, and *the aggregation level of intelligence* (whether it is an aggregation or composition of several entities). The degree of intelligence of smart products depends on how well they *handle information*, *identify and solve problems*, and *make appropriate decisions* (Figure 1.10).

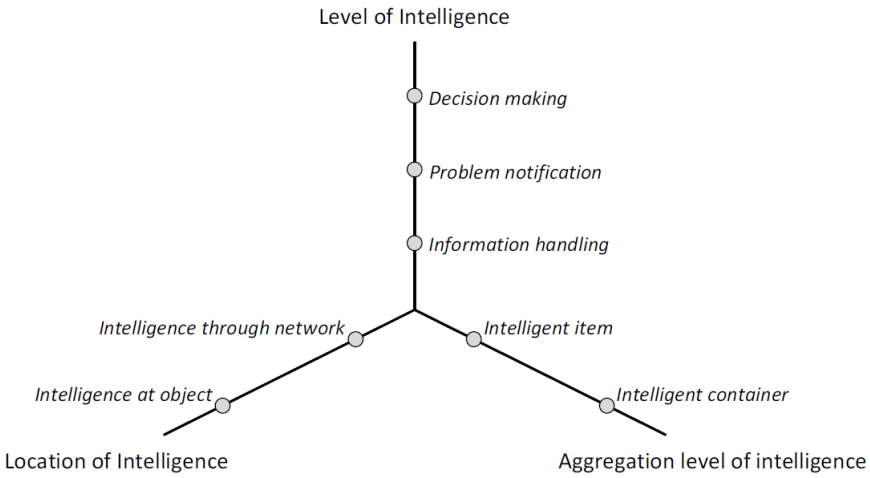


Figure 1.10 Classification model of smart products (Meyer et al., 2008)

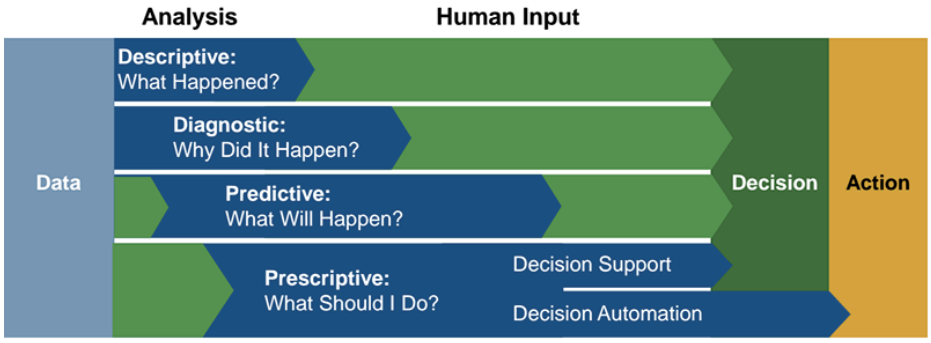


Figure 1.11 Gartner analytics capability framework (Steenstrup et al., 2014)

The increasing use of sensors within smart products provides the data needed for intelligence. Machine learning and data analytics provides the tools and technologies needed to increase the degree of intelligence. Data analytics has emerged as a generalized method for processing information to answer the questions of "what has happened" (*descriptive analytics*), "why did it happen" (*diagnostic analytics*), "what will happen" (*predictive analytics*), and "what should we do" (*prescriptive analytics*),

using combined analytical, statistical, and machine learning techniques. As shown in Figure 1.11, as the capability of analytics increases, human input decreases. But, how does the analytics capability actually connect to the physical parts within the smart product to fulfill the intelligent functions?

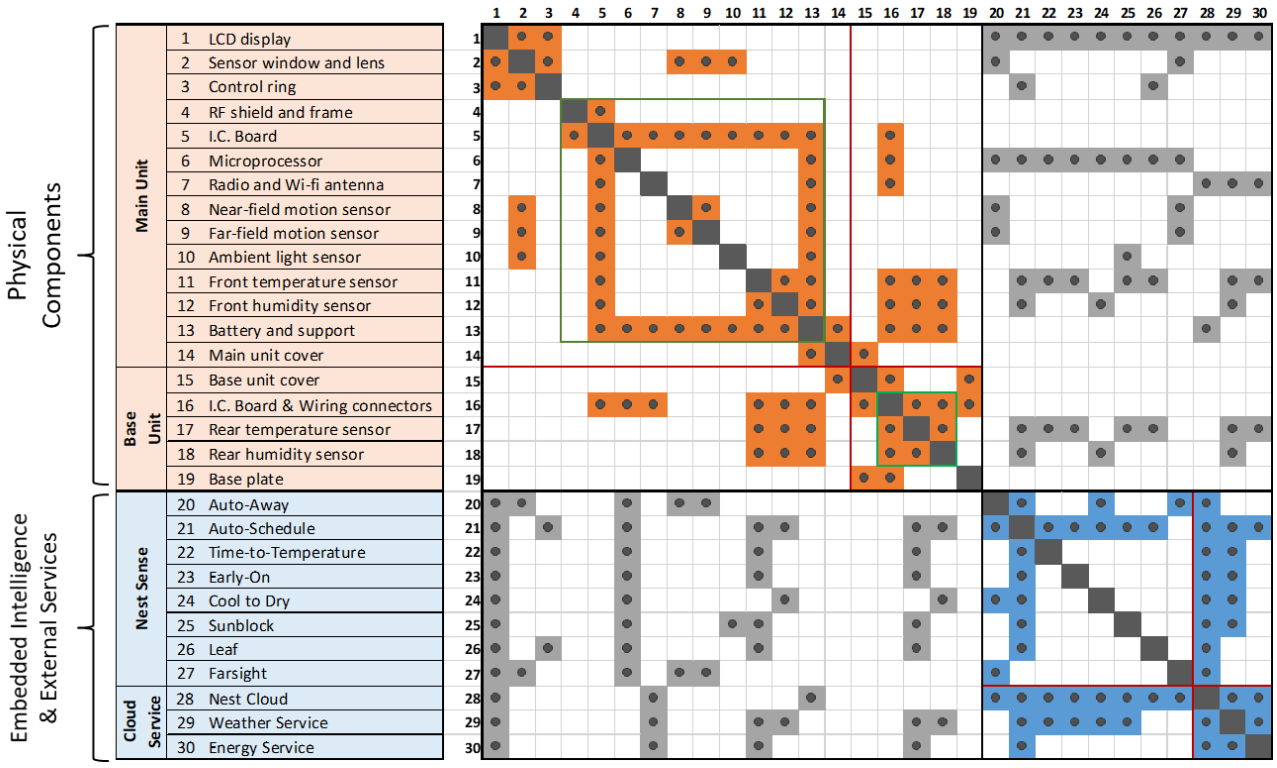


Figure 1.12 The component decomposition of the Nest Thermostat

Figure 1.12 shows a design structure matrix (DSM) representation of the interactions among the Nest Thermostat's physical components, the embedded analytics features, and the cloud-based analytics services. Each dot in the matrix indicates there is at least one of the five dependencies (*Spatial*, *Structural*, *Energy*, *Material*, and *Information*) that may present between paired components. While spatial, structural, energy, and material dependencies are mainly related to physical components (Sosa et al., 2003), the information dependency is needed for both physical components and analytics features. For physical components, it is required to realize functional requirements that transfer signals or controls; for analytics features, it is completely data and information based.

We observed five roles of the analytics features from this DSM decomposition, which illustrate how

the analytics features (along with the physical components) do enable the product intelligence:

1) *An analytics model is a Part/Component of another analytics model to fulfil certain functions.*
   Example. The *Time-to-Temperature* feature is to predict when the target temperature will be reached. Understanding how quickly a house is warmed up or cooled down, under varying weather conditions, the thermostat can capture the thermal dynamics inside the house. This in turn helps the Auto-Schedule feature generate an optimal schedule.
2) *An analytics model is a Part/Component of the product system to fulfill certain analytical functions.*
   Example. The *Farsight* feature allows the thermostat intelligently displaying different contents by detecting how far the user is standing from the thermostat. This feature is embedded in the thermostat and the scope of its function is restricted only to the host product.
3) *An analytics model is a Service of the product system to fulfil certain remote analytical functions.*
   Example. The *Energy Service* resided in the *Nest Cloud* can tune the temperature on each individual Nest Thermostat around energy rush hours that happen on extreme weather days. Each Nest Thermostat then pre-cools or heats the building by using more energy before a rush hour, in order to prevent possible blackouts. The scope of this function is for all the connected thermostats.
4) *An analytics model is a Configuration Rule of the product system.*
   Example. Once a Nest Thermostat is deployed in service, it starts to learn and adapt. The Auto-Schedule feature generates unique schedule rules for each individual thermostat. Other relevant features are also dynamically enabled, disabled, and reconfigured based on the actual field conditions. As a result, each Nest Thermostat behaves differently.
5) *An analytics model itself is a standalone Product that needs be developed and maintained.*
   Example. The first-generation Auto-Schedule feature was developed based on simulation data. Three years later, an *Enhanced Auto-Schedule* feature was released (Nest, 2014). This upgrade is a result of utilizing the accumulated actual usage data collected from many houses across different climate regions, thus more accurately captures thermal dynamics and users' behaviors.

1.3 Problems and Challenges

1.3.1 "Data-Driven" Product Development Paradigm

Data has become a critical factor that drives the smart products development, operations, and

improvement activities related to the products themselves and their ecosystems. The two data-centered product design paradigms, *data-informed* and *data-driven*, have been both seen regarding the Nest Thermostat's Auto-Schedule development. On one side, the evolution of the auto-schedule feature shows that how the thermostat usage data had been utilized to measure the product performance and incrementally improve the product experience. On the other side, the thermostat's adaptation to individual house and the user shows that how the environmental data and the user behavior data had been used as the "materials" to produce individual predictive models and to prescribe individual heating/cooling schedules to control the corresponding HVAC systems.

Table 1.3 Data-centered design paradigms (Patil, 2012; Dhar, 2013; Pavliscak, 2015)

Design Paradigm	***Roles of Data***	*Roles of Data Analytics*
Data-Informed Product Design	Data is utilized and analyzed to improve the process to develop and operate products. It is used to • Reveal patterns and trends to drive innovation; • Incrementally improve the product experience; • Measure the performance for feedbacks.	Data analytics is embedded in other processes to improve those processes
Data-Driven Product Design	Data is utilized as materials to create products. It is used to • Produce artificial intelligence models based on insights gained from the data; • Continuously improve the artificial intelligence based on more data; • Generate new data for further processing.	Data analytics is the focused process to produce data products

As shown in Table 1.3, the roles of data analytics played in these two scenarios are different: in the first case, the data analytics is embedded inside the product development and operations processes; in the second case, the data analytics is actually the focused "engineering" process to produce data-based products. More specifically, as indicated in previous observations, the Nest Thermostat's embedded data analytics features can be seen as product parts/components rather than operational functions. This is because the analytics features adapt for each installed instance rather than aggregate functions for a fleet of products. Consequently, data analytics is no longer an operational process, but rather, a product

development activity that introduces new product features. That is to say, the Nest development team needed to co-develop the physical architecture (e.g. HVAC control) and the data architecture (e.g. a home model) to achieve optimal solutions for energy savings. This is consistent with Mühlhäuser's argument – the product and corresponding smartness need to be co-constructed.

That implies the product engineers (mechanical and electrical engineers) need to work closely with software engineers and data scientists (if this role can be decoupled from or newly introduced into the development team) to co-develop the product in order to design more "software-driven" even "data-driven" features, as shown in Figure 1.13. Ideally, they even need to collectively formulate the problem, explore, screen, and evaluate the potential concepts, and eventually select one or more optimal concepts to finalize the product specification, during the early product design stages. This has several implications that would challenge the traditional product development paradigm.

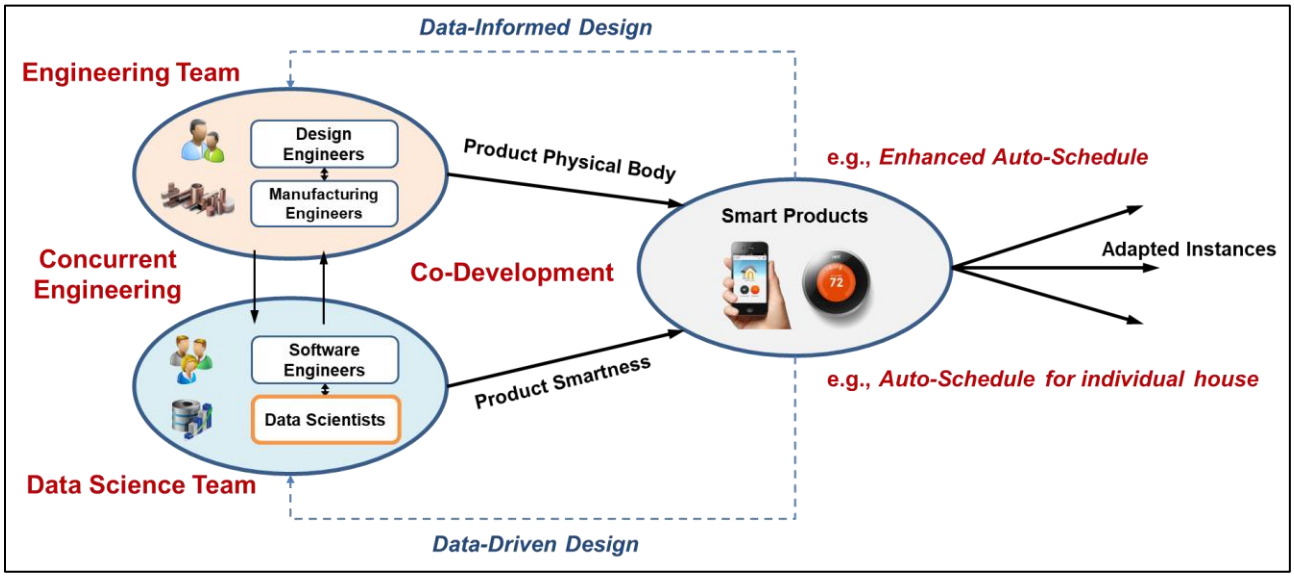


Figure 1.13 A multidisciplinary team to develop smart products

1.3.2 Challenge 1: *Data Analytics Engineering*

The first question encountered by the product development team for product intelligence design is:

1) *How to decouple the data for the data-informed process improvement purpose and for the data-driven feature development purpose?*

This is nontrivial because a product may consist of hundreds and thousands (for complex products, even more than ten thousand) of parts and each part is associated with numerous supporting data (e.g. engineering drawing, test data, maintenance history) at different life cycle stages, as shown in Figure 1.14. We need to understand what data is used for process improvement and what data is used for creating data analytics models as the product's smart features.

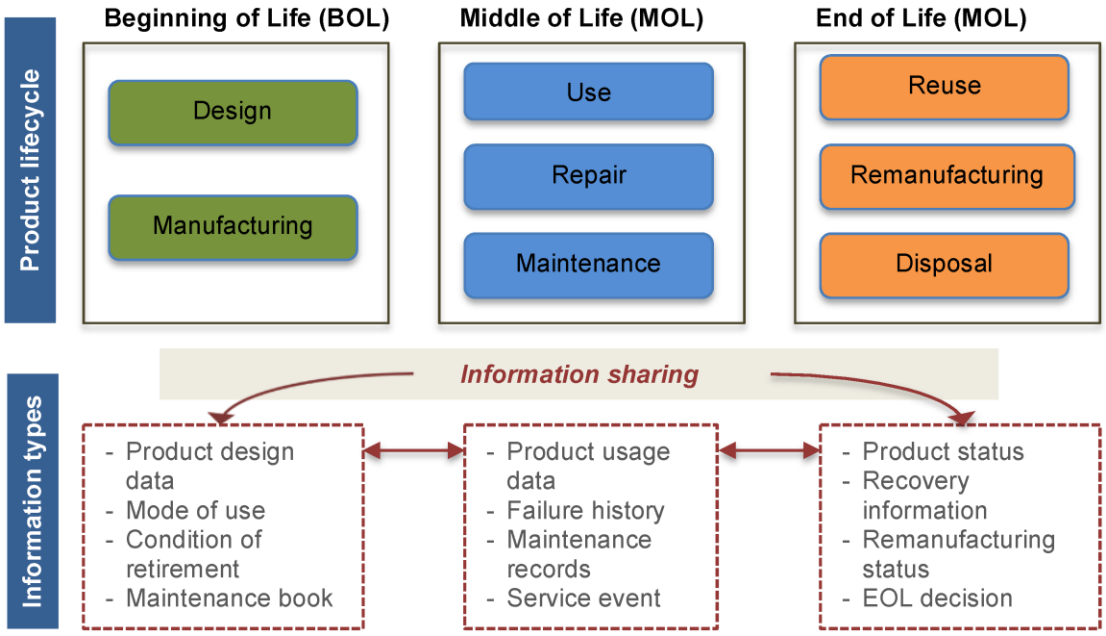


Figure 1.14 Product lifecycle management and information sharing (Yoo et al., 2016)

All the data and their connections need to be recorded and tracked to keep the product in safe usage, effective maintenance, disposal and recycling when it reaches the end-of-life. Enterprise systems like PLM/ERP are recording product-related data from almost all the processes of an organization such as product and process design, material planning and control, assembly, scheduling, maintenance, and recycling (Roy et al., 2014). Real-time sensor data from products (e.g., GPS sensor in a mobile phone) in use is enriching this dataset such that provides a wealth of insights into real usage conditions and the

actual product requirements, product failure information, customer needs and habits, and other real-world data (CIMdata, 2012a;2012b).

To answer the data decoupling question, let's extract the Nest Thermostat's auto-schedule's decision-making process and its data dependency from the DSM shown in Figure 1.12. When we have established the data/decision hierarchy, we found that the auto-schedule can aggregate the decisions from other primitive predictive models: *Auto-Away* (based on algorithms that interpret occupancy sensor data and provide a confidence determination of whether or not the occupants are away from the home), *Time-to-Temperature* (a prediction of when the target temperature will be reached), *Sunblock* (using the built-in light sensor to track the sun's patterns and its temperature sensors to detect the heat spikes that occur in direct sunlight; it also considers sunrise and sunset schedule for the user's location), etc. This leads to a new question: can data analytics models be seen as another type of products.

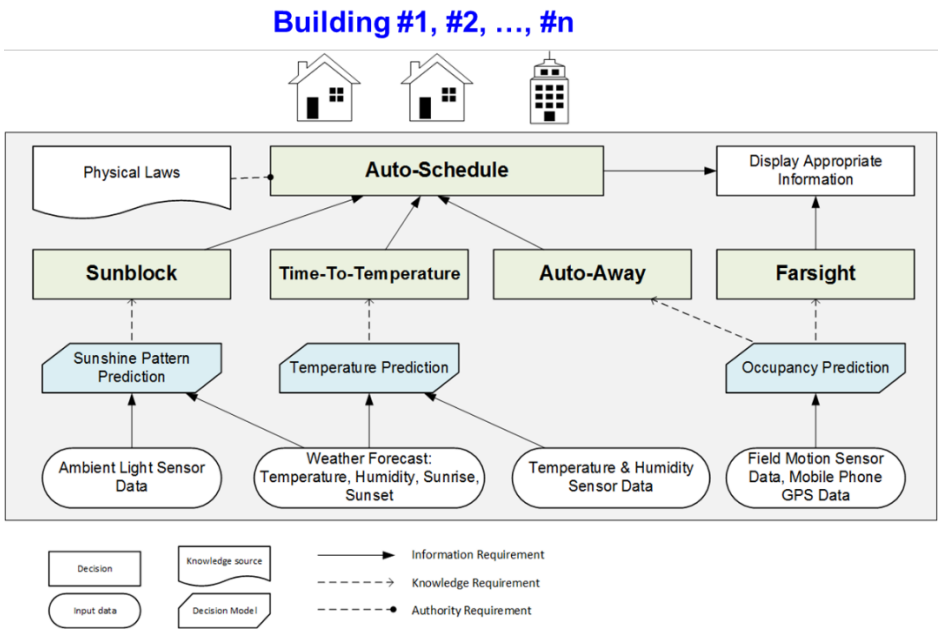


Figure 1.15 The decision and data dependency of the auto-schedule feature[4]

[4] Note: We use the DMN (Decision Model and Notation) convention to represent decision-making logics in this dissertation. We will explain the DMN symbols in details in Chapter 3.

As a physical product typically consists of subassemblies, and each subassembly may be composed of several individual parts, the product is usually represented as an assembly tree. Individual parts are produced from certain kinds of raw materials, and could be supplied by various manufacturers. Parts with common functions are standardized for easier reuse, interchange, mass production, and mass customization. A part might be used as many instances in the same product or in a different product and may have variations.

Considering data as the raw material, data analytics can be seen as a production process for producing data products, delivering data or delivering results based on data. The resultant analytics model is then an information-processing unit, taking data as inputs, and generating certain level of decisions. In this sense, an analytics model could carry similar hierarchical characteristics as a physical product. Then, any analytics model is possibly abstracted as shown in Figure 1.16, a master model can consist of a set of component models. Each component model may be composed of several unit models. Thus, the master model can be represented as an assembly tree of component models and unit models. Models with common functions can be standardized for easier reuse, interchange, and composition. A model might be used repeatedly in the same master model or in a different master model and may have variations.

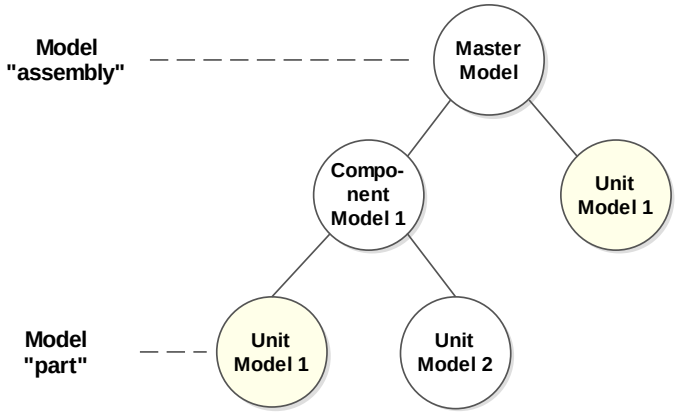


Figure 1.16 An analytics model assembly tree (Li et al., 2015a)

If this assumption holds for the intelligence of smart products in general, the design of analytics models that implement the intelligence can be decomposed into designs of smaller modules that are then integrated later. Hence, it transforms the smart products development into a transdisciplinary collaboration between the traditional engineering team and a data science team that dedicates to the development of data-based products. A data requirement specification would also be available for the data products development. But this raises two questions:

2) *How to conceptualize a smart product so that the smart features and their associative physical components can be modularly decomposed in order for concurrent engineering?*
3) *What knowledge should we elicit for the involved physical products, data products, and the criteria or configuration rules so as to compose them over different product lifecycle stages?*

1.3.3 Challenge 2: *Co-Development of Physical Products and Data Analytics*

It has been already recognized that creating mechatronics requires transdisciplinary collaboration across mechatronics, software, and service domains, in which each domain has its own unique design and development process. Most engineering process models – mechanical, software, service – focus only on individual domain rather than consider other domains in a system perspective. The interaction between different processes which contribute to the creation of the final product is usually not addressed in these discipline-specific process models. An effective process to support transdisciplinary collaboration at the full lifecycle spectrum should take the systems engineering approach and support multiple modes (sequential, spiral, and V-form) (Gericke and Blessing, 2011). Therefore, a higher level of process abstraction to be discipline-independent is necessary.

Now, the development of smart products involves a new discipline, Data Science. There have been many efforts in the data science community to formalize the data analytics processes. The term of “Data Product” has been proposed by several data science pioneers. A narrow definition of data product is to

represent a concrete component that facilitates the end goal analysis through the processing of data (Patil, 2012). This means a data product can be raw data or intermediate datasets. However, analytics models are also the results of processing data indeed, i.e. they are also data products. Thought of this way, data analytics can be seen as an engineering process, *Data Products Development (DPD)*. It produces software-like but data-centered products and tools (e.g. data processing pipelines, statistics and machine learning algorithms, and mathematical analytics models). However, the data analytics process is different from the traditional software development process because of the requirement to monitor and tune the model in short iterations and the fact that it is difficult for data scientists to know *a priori* what will be found when "exploring the data" (Saltz, 2015). In short, data analytics creates new knowledge while software program automates known knowledge.

On the other side, the traditional product development process can also be seen as an information-processing system or a decision production system, in which a network of stakeholders carries out various activities to process the development information, formulating specifications, concepts, and design details (Ulrich and Eppinger, 2012). The process concludes when all the information required has been created and communicated, as well as when the key decisions have been made within the project time and budget constraints (Herrmann and Schmidt, 2002; Krishnan and Ulrich, 2011). This perspective implies there have been data analytics tasks embedded within the product development process. Now the question is, how to decouple the tasks for data scientists from a product development process? In another words,

4) *What are the key, specialized tasks that the engineers in a physical product development team and the data scientists in a data products development team need to conduct?*

Accordingly, the following questions should be addressed as well:

5) *Which tasks need to be coordinated across the two team groups;*

6) *When and what information needs to be exchanged between the two groups to collectively achieve the product development; and*

7) *What are the patterns and characteristics of their interactions?*

1.3.4 Challenge 3: *Synchronize Lifecycles*

The other common and inevitable problem for any products is the engineering change, which is challenging to be handled effectively and efficiently (Jarratt et al., 2011). The ever-evolving nature of smart products need to incorporate changeability into the product architecture to enable four characteristics including the flexibility, agility, adaptability, and robustness (Friche and Schulz 2005). *Modularity* (or *Encapsulation*) has been recognized as a key principle to realize these four characteristics, by clustering the system's functions into various modules while minimizing the coupling among the modules and maximizing the cohesion within the modules. The concept of modularity has been well researched and accepted by the product design and manufacturing communities to handle the product and process configuration/reconfiguration issues in the past (Suh, 1990; Ulrich, 1995; Ishii, 1997). This concept has also been extended in design of service (Voss and Hsuan, 2009), supply chain design (Bask et al., 2009), and also in design of platform-based product families to support mass customization business model (Jiao et al., 1998; Simpson et al. 2001; Simpson et al., 2005; Zha and Sriram, 2006).

Product configuration involves combining variations of modular, configurable physical products, software product lines, and configurable services (Quéva et al., 2011). The configuration space is huge if all these domains are taken into consideration. For example, an automobile has very high level of reconfigurable product varieties. A modern car such as the Jaguar Land Rover Range Rover Sport has buildable combinations at the magnitude of 10^{21} (Batchelor and Andersen, 2012). The growth of software-driven and data-driven technologies such as telematics, infotainment and telecommunications

systems in cars implies that the way in which the cars can be configured will exponentially increase. While the growth of software-related features does not necessarily mean variation of hardware, the supply chain management becomes increasingly complex.

In addition, product reconfiguration is not subject to a single technology lifecycle, but to the multiple lifecycles of all the major systems and subsystems. The number of valid combinations is not a static number but a dynamic one that varies over time as the technical contents of the product change, or the regions and markets alter (Batchelor and Andersen, 2012). Reconfiguration is inevitable in the after-sales lifecycle of configurable products, particularly considering the high frequency of software update and new data-driven service offerings.

Many lifecycle concepts – e.g., application lifecycle management (ALM), product line engineering (PLE), model lifecycle management (MLM) – have been proposed to address issues within an individual domain or inter-domains (Fisher et al., 2014; Rowell and Ballou, 2014). However, none of them has addressed the issues to incorporate the new discipline, data science, and the new experts, data scientists, into the product development team.

1.3.5 Challenge 4: *Standards and Tools for Data Interoperability and Collaboration*

Product Lifecycle Management (PLM) systems provide shared platforms for creating, managing, and disseminating product-related information across the extended enterprises (Ameri and Dutta, 2005; Grieves, 2005). Traditional PLM systems are mainly based on relational database. The large volume, fast velocity, and high variety of sensor data of smart products are easily overwhelming the existing database. Modeling, mapping and synchronizing different data schemas are very costly. Furthermore, successful collaboration among mechanical/electrical engineers, software engineers, data scientists, and business experts is a big challenge, because it is often difficult to communicate ideas across multiple disciplines and to integrate the digital representations of those ideas among heterogeneous tools and

information systems (Li et al., 2015b).

The usual approach to address these issues involves standards. In engineering world, plethora of industrial standards are available and ranged from product design, plant control, SCADA, manufacturing operations management, business logistics, to inter-company collaboration (Gifford et al., 2006). These include ISO 10303 STEP (Li and Roy, 2014) for product models, ISA-95/B2MML[5] and OAGIS[6] for manufacturing operations, MIMOSA[7] for operations and maintenances, MTConnect[8] for numerically controlled machine tools, EPCglobal[9] for radio frequency identification (RFID), OASIS PLCS[10] for product lifecycle support, and SCOR[11] for supply chains, to name a few. While many of these standards have been available for decades, none of them supports all of the interdisciplinary information needed. These standards have not been sufficiently adapted for smart products development that incorporating data and information as commodity products.

On the other hand, new machine learning algorithms and data analytics techniques are fast emerging. There can be an inability of big data processing within manufacturing firms due to the limitations of IT resources within those firms (Sun et al., 2017). Furthermore, few manufacturing experts are familiar with modern big data analytics techniques. While data-analytics-related standards such as PMML[12] (Predictive Model Markup Language) and CRISP-DM (CRoss-Industry Standard Process for Data Mining) (Shearer, 2000) are available and well-known in the data analytics community, these standards remain largely underutilized by the manufacturing industry today, and are not supported by the current PLM systems. This raises another question:

---

[5] Business To Manufacturing Markup Language, http://www.mesa.org/en/B2MML.asp
[6] Open Applications Group, http://www.oagi.org/
[7] Operations and Maintenance Information Open System Alliance, http://www.mimosa.org/
[8] MTConnect Institute, http://www.mtconnect.org/
[9] Electronic Product Code, http://www.gs1.org/epcglobal
[10] OASIS Product Life Cycle Support, https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=plcs
[11] The Supply Chain Operations Reference model, http://www.apics.org/sites/apics-supply-chain-council/frameworks/scor
[12] Data Mining Group, http://dmg.org/

8) *How can we harmonize the information standards in traditional manufacturing and new data science domains to address the interoperability issues and develop the next generation of tools to facilitate smart products innovation?*

1.4 Research Objectives

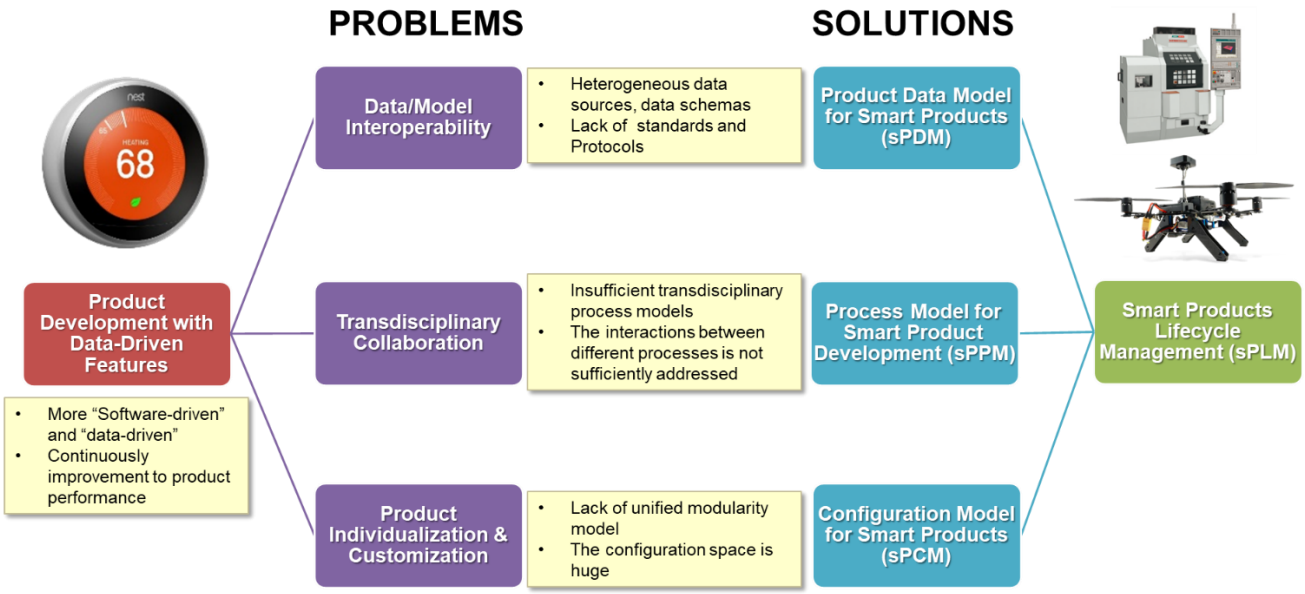


Figure 1.17 Research problems and target solutions (The images of the thermostat, the quadcopter, and the CNC machine are credited to Nest Labs, Intel Corporation, and DMG MORI USA)

To address the four challenges and the eight questions, we define that the objective of this research is to **develop a formal modeling framework for smart products development involving co-development of physical components and data-driven features.** First, the framework shall address the interoperability by uniformly representing and composing physical products created by engineers and analytics models created by data scientists. Second, the framework shall help support and optimize the information flows and interactions between engineers and data scientists so that they can collectively conceptualize solutions during the product development. Third, the framework addresses the modular design issues of analytics models so as to facilitate the dynamic reconfiguration and adaptability of smart products, given the frequent engineering changes of data analytics models. Fourth, the framework

shall support existing, well-adopted open/industrial standards for across-domain information modeling, data/information exchange, and team collaboration.

As shown in Figure 1.17, three models – sPDM, sPPM, and sPCM – will be developed to capture the abovementioned requirements. And, these three models will formulate a Smart Products Lifecycle Management (sPLM) framework to achieve the research objective. The Nest Thermostat example will be continuously used to discuss our motivation whenever necessary. And, a CNC machining center and an unmanned aircraft system (UAS) will be used as case studies to develop the models and implement the proof-of-concept platform.

The innovation and foundation of this research is **viewing data analytics as a Data Products Development process** based on the understanding of how data analytics is implementing to the increasing intelligence of mechatronics. The contribution of this research is to **expand the "Product Lifecycle Management (PLM)" concept traditionally for physical products to data products so that we can incorporate into the traditional physical product development methodologies with the ongoing efforts done in the data science community**. As a result, a Smart Products Lifecycle Management (sPLM) framework is conceptualized and is implemented based on industrial and open standards for validation.

The author of this research believes:

- The systematic methodology to develop smart products shall be different from that for the traditional "dumb" products due to the new *Intelligence* dimension. Tackling this problem will contribute to the development of next generation of product innovation platform.
- Data can be seen as raw materials to produce data products. If common characteristics exist between physical products and data products, the long-established product design and development methods (e.g. product lifecycle management, modular design, mass customization, lean manufacturing) can then be adapted and applied to data products development.
- The independent advance of formal methodology to data products development shall in turn

complement the traditional theoretical methods for product design and development so as to support the smart products design paradigm. Thus, collaborating with data science researchers is necessary.

- The research on smart products development shall be conducted at the system scale and at the lifecycle spectrum rather than focused on a single component and a single lifecycle stage, as smart products are complex systems in nature.

1.5 Chapter Structure

In following chapters, each chapter starts with the conceptualization of the individual topic, followed by demonstration of how the proof-of-concept can be implemented, and then elaborates how the concept can be applied using examples in development of smart products and boarder manufacturing systems. The remaining of this dissertation is organized as follows (see Figure 1.18):

Chapter 2 reviews the exiting theoretical frameworks and methods related to product data/model interoperability, transdisciplinary collaboration, and product individualization, as well as the techniques for product/process information modeling and analysis. We then contextualize a *Smart Products Hypercube* information model to project a smart product's design space. This high-dimensional design space is then decomposed using Tensor analysis and three research spaces are identified: (1) issues related to productization, modularization, and lifecycle management of analytics models; (2) unified information modeling to support composition and configuration of smart product components; and (3) integrated process for co-development of physical components and data analytics models.

Chapter 3 discusses the requirement and structure representations of analytics models taking two perspectives – product and decision-making – into consideration. Then we discuss the systematic method to tackle the issues related to the modular architecture design of analytics models and the modularity quantification of such an architecture. A polynomial regression model for CNC machining center energy prediction is used to illustrate a proof-of-concept implementation of *Analytics Model*

Lifecycle Management. This chapter is related to all four challenges discussed previously.

Chapter 4[13] proposes a *Smart Component* data model that incorporates analytics models as "parts" or "services" of products in their master records. The smart model unifies the digital representation of physical components and analytics models so these two heterogeneous entities can be composed using appropriate configuration rules. This allows a component with both physical parts and unit analytics models can be modularized, composed, reused, traced, maintained, and replaced on demand. The bill-of-materials (BOM) concept is extended to have both physical components and analytics models to represent a complete part list of a smart product. The proof-of-concept implementation is elaborated using a smart CNC machining center case study. We validated the smart component model by using it to develop a modular, hybrid sustainability assessment system where both knowledge-based models and data-driven models can be accommodated for and combined at different abstract levels in assessing a product's sustainability. This chapter is primarily related to the challenge 3 and 4 discussed previously.

Chapter 5 proposes an integrated process model, *NPD³ model*, for new product development with data-driven features. The chapter revisits the existing process models for physical product development, software development, and data analytics, since each one has prescribed the common activities used in many practical projects. The standard steps and activities prescribed in these existing models provide an initial view of how engineers or data scientists individually work. The potential collaboration points are then hypothesized by aligning and comparing these models. This analysis helps to derive an initial integrated model that for both engineers and data scientists. The hypothesized model is applied to a real-world smart product development case, and an information decomposition framework is then developed

[13] **Note:** Chapter 3 and Chapter 4 have greatly extended and augmented our prior work regarding modular design of data-driven analytics models, the development of smart component data model, and the implementation of data analytics information models in PLM, which were initially published in ASME (American Society of Mechanical Engineers) and IEEE (Institute of Electrical and Electronics Engineers) conferences (Li et al., 2015a; 2015b; 2017a; 2017b).

to qualitatively categorize the observations of the interaction patterns within the case study, which leads us to achieve a theoretical framework for presenting the detailed interaction contents of the information flows. The interaction patterns and information contents complement the initial view of the integrated model that depicts the high-level key tasks and information flows. This chapter is primarily related to the challenge 2 and 4 discussed previously.

Chapter 6 reports a case study regarding the development of a specialized *Product Lifecycle Management system for Unmanned Aircraft System (UAS)*. This application is used to validate the usability of the proposed sPLM framework. The smart product hypercube model is applied to decompose and represent the UAS hardware, autonomy-enabling functions, and the underlying data architecture. The NPD^3 process model is used to guide the co-development of the physical components and the analytics capabilities of a smart UAS. The sPLM implementation provides the UAS development team a collaborative environment and data repository to facilitate effective data/information exchange, visual communication, and traceable decision-making.

Chapter 7 summarizes the smart products lifecycle management framework and recap the related topics developed in this research. We then conclude the merits of the research, the findings and achievements. Finally, we review the research limitations and propose potential future research.

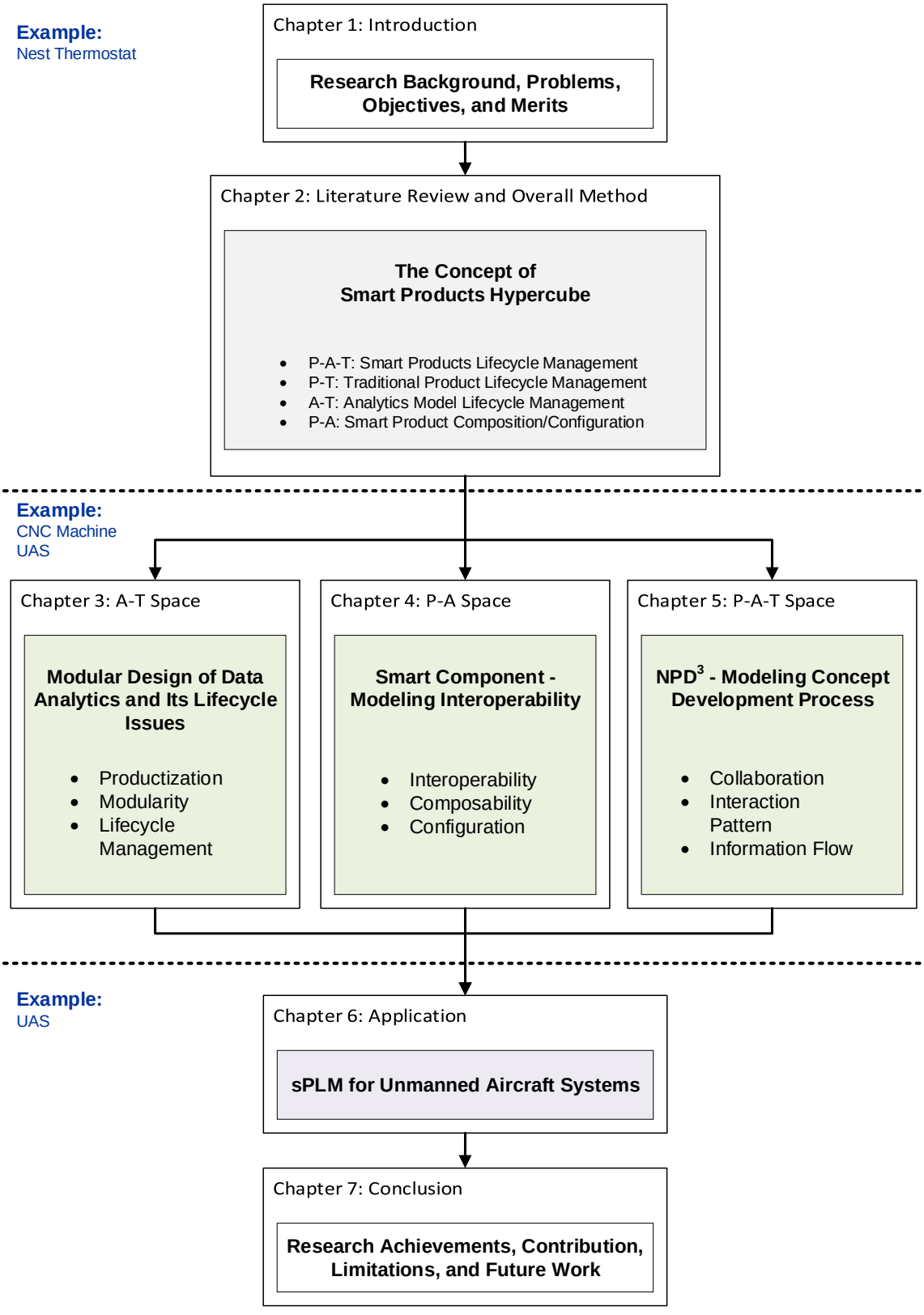
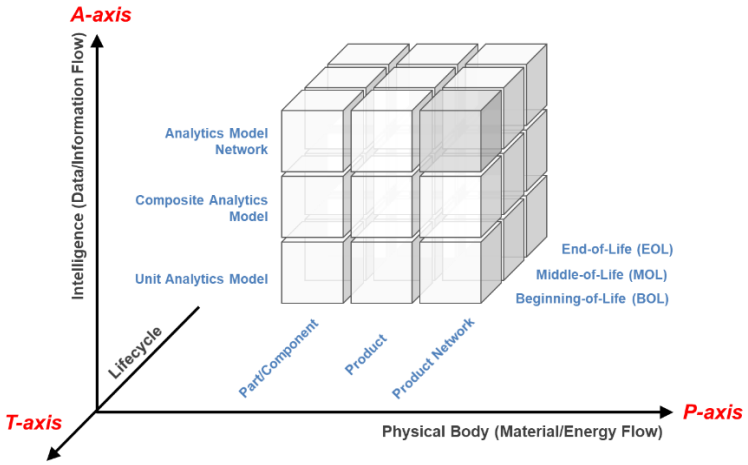


Figure 1.18 The chapter structure of the dissertation

Chapter 2

The Concept of "Smart Products Hypercube"



The components constructing a smart product can be functionally classified into three categories: *Physical components*, *Smart components*, and *Connectivity components* (Porter and Heppelmann, 2014; 2015). In term of this classification, *Physical components* are mechanical and electrical parts that form the product body; *Smart components* include sensors, microprocessors, data storage, controls, and digital user interface, so that the product can collect data, process information, and react to the environment it resides; and *Connectivity components* are related to ports, antennae, protocols, and networks that enable communication between the product and the product cloud.

The information and knowledge to represent these functional components and the process activities to develop them is coupled in hardware, software, service domains, and now one more domain, data science. For example, sensors are also physical components. The sensor shape, size, location, interfaces with other physical components, and the sensors' performance specifications have to be carefully considered to accommodate the overall product design. Furthermore, the smartness of a product can be designed within the product and can also be distributed across the cloud, which implies the connectivity

components also contribute to the smartness. Since data analytics and computation have become driving forces to the increasing intelligence of smart products, the Cyber-Physical Systems (CPS) concept is perhaps more suitable to classify the components of smart products with two higher abstractions: the abstraction related to *modeling physical processes* and the abstraction related to *modeling processes of transforming data*. The key challenge is to conjoin these two abstractions.

In next sections, we first review the exiting theoretical frameworks and methods related to product data/model interoperability, transdisciplinary collaboration, and product customization, as well as the techniques for product/process information modeling and analysis. We then hypothesize a smart product can be seen as appropriate configurations of physical components and analytics models, therefore, we contextualize the design space for smart products as a hypercube that projects the physical components design, analytics models design, and time-based lifecycle stages on three orthogonal dimensions. This formulates the research domains for the remaining parts of this dissertation and leads to the development of a next generation PLM system, named *Smart Products Lifecycle Management (sPLM)*.

2.1 Literature Review

2.1.1 Product Information Models for Data/Model Interoperability

Information Models for Physical Products

The efforts to standardize physical product information have last for three decades. One successful standard is the ISO 10303 series, which is formally entitled “Industrial automation systems and integration – Product data representation and exchange”, and is also informally known as STEP (STandard for the Exchange of Product data). The ISO STEP standard is originally designed for the representation of product geometric information but has been intensively extended for computer-aided design (CAD), engineering (CAE), and manufacturing (CAM).

The ISO STEP standard is designed as modules, and individual module is called "Part" and "Application Protocol (AP)." Each Part defines a specific scope and each AP is for implementation in industry to address specific products and processes. Table 2.1 lists the APs that are commonly used in manufacturing information systems. A detailed review can be seen in Li and Roy (2014).

Table 2.1 ISO STEP standards for industrial applications

Category	***Application Protocol***
Generic Standards	AP203 – Configuration controlled 3D designs of mechanical parts and assemblies
	AP235 – Materials information for the design and verification of products
Requirement and Concept Standards	AP233 – Systems engineering data representation
Analysis Standards	AP209 – Composite and metal structural analysis and related design
	AP237 – Computational fluid dynamics
Detailed Design/Bill of Materials Standards	AP210 – Electronic assembly, interconnect and packaging design
	AP214 – Core data for automotive mechanical design processes
	AP212 – Electrotechnical design and installation
	AP232 – Technical data packaging: core information and exchange
Manufacturing (Make and Buy) Standards	AP219 – Dimensional Inspection
	AP224 – Mechanical product definition for process planning using machining features
	AP238 – Computer numerical controllers (STEP-NC)
	AP240 – Process Plans for machined products
Life cycle Support Standards	AP239 – Product lifecycle support (PLCS)

The modular design of the ISO STEP standard allows multiple APs to be collectively used for satisfying the product lifecycle information exchange needs across industrial applications. However, non-geometry information is not readily supported by the native ISO STEP translators of many commercial CAD tools (Li and Roy, 2014). Furthermore, ISO STEP lacks support to sustainability-related information, which is critical in the contexts of Sustainable Manufacturing and Smart Manufacturing. In order to overcome these issues, NIST proposed the Core Product Model (CPM) as another abstract model for product data formalization. The CPM represents a product's function, form and behavior, its physical and functional decompositions, and the relationships among these concepts (Figure 2.1). It provides a base-level product model that is open, non-proprietary, generic, extensible, independent of any one product development process and capable of capturing the full engineering context commonly shared in process development (Fenves 2001).

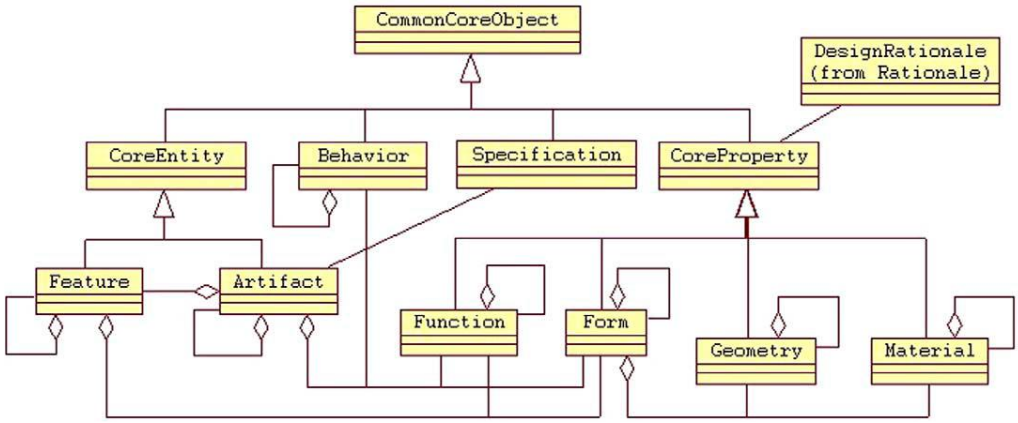


Figure 2.1 The NIST CPM Model (Sudarsan et al., 2005)

The CPM has several extensions: (1) the *Open Assembly Model (OAM)* defines a system level conceptual model and the associated hierarchical assembly relationships; (2) the *Design-Analysis Integration Model (DAIM)* defines a Master Model of the product and a series abstractions called Functional Models (one for each domain-specific aspect of the product) and two transformations, called idealization and mapping, between each master model and each functional model; and (3) *Product Family Evolution Model (PFEM)* extends the representation to families of products and their components; it also extends design rationale to capture the rationale for the evolution of the families. These models altogether found a product information-modeling framework to support the full range of PLM information needs (Sudarsan et al., 2005), and they have been proved performing well on physical products.

For example, Barbau et al. (2012) proposed an approach to enable the translation from data defined in ISO STEP EXPRESS language into OWL (Ontology Web Language). Geometric information of a product represented by STEP AP203 is translated into OWL first, then the beyond-geometry information represented by CPM is added to the ontology to generate a semantically enriched ontology. This new product model is named OntoSTEP. Sarigecili et al. (2014) presented how to interpret the Geometric Dimensioning and Tolerancing (GD&T) specifications in STEP for tolerance analysis by utilizing the

OntoSTEP model. The CPM model itself and the extended version, OntoSTEP model, show potentials for representing and exchanging product information and relevant life cycle information in a design for sustainability scenario. Li and Roy (2014) proposed an ISO STEP and CPM hybrid framework to compose core product information and sustainability-related information into a unified information model to support data sharing and exchange between computer-aided design tools and life cycle assessment tools. While these application efforts had enriched the CPM model under various contexts, the extensions of CPM to services, software products, data products, however, have not been reported to the author's knowledge.

Recently, product non-geometric information (e.g. geometric dimensioning and tolerancing) has been generalized as product and manufacturing information (PMI), and been standardized as a new ISO STEP application protocol, AP242 (ISO, 2014). This increases ISO STEP's capability in representation of digital product definition, also known as model-based definition (MBD) (Briggs et al. 2010; Quintana et al. 2010; Lubell et al. 2012).

The ISO STEP AP242, being titled "Managed Model Based 3D Engineering", merges two widely used STEP standards: *AP203* (for Configuration Controlled 3D Design) and *AP214* (for Core data for automotive mechanical design processes). One goal of the AP242 is to support the information modeling for multi-disciplinary products. Therefore, it proposes a comprehensive business object (BO) model for capturing a product's requirement specification, configuration, 3D geometry, machining features, manufacturing specifications and annotations, manufacturing process plan, spatial relationship, material and composite structure, as well as supporting essential product data management elements (see Figure 2.2). Furthermore, an XML schema derived from the BO model can be used as an implementation model to support data exchange related to product data management systems and associated services. The new BO schema also provides the *External References* capability to link any XML-based model as digital

representation of a product's master data. This provides the feasibility to extend AP242 to capture a product metadata defined by another standard as its digital representation.

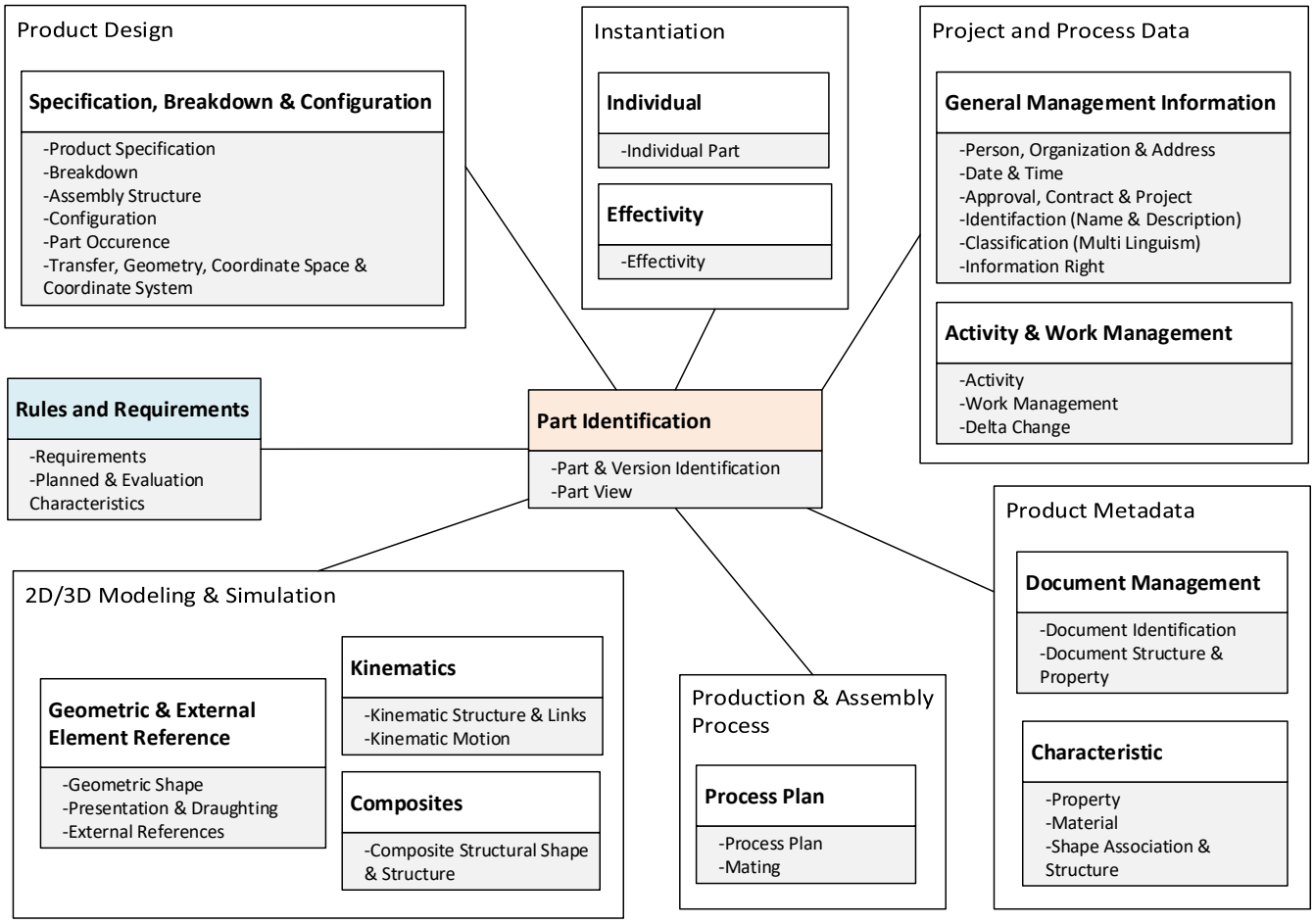


Figure 2.2 ISO STEP AP242 business object model capabilities (Adapted from ISO, 2014)

Information Models for Data Analytics

In data analytics community, PMML is perhaps the most prominent standard for analytics model representation. PMML is originated from the Data Mining Group[14] (DMG) and it uses XML to represent mining models. The structures of analytics models are described by an XML Schema. PMML has been widely supported by mainstream data analytics platforms including commercial software and

[14] Data Mining Group, http://dmg.org/

open-source packages. Research in academia and industry has been very active in areas of: (1) In-database analytics; (2) Knowledge extraction for business rules management systems; (3) Scoring engines and big data platforms; and (4) Analytics model management (Li et al., 2015a; 2015b).

PMML supports popular predictive models such as classification, regression, clustering, and association rules. PMML has the potential to support the analytics model productization idea, because of its structured schema, openness, wide industrial adoption, and multi-model support. PMML separates the development and deployment processes of predictive models, and enables interchange of the models among different data analytics tools and environments. However, there are critical challenges in using PMML to represent analytics models in a similar fashion as the representation of physical products. First, PMML is designed primarily for representing model structures, neither data nor management functions. Therefore, for any data fusion and data management, we have to rely on external, specific, workflow-based data analytics tools. Second, PMML provides no support for external document references; consequently, it puts limits on the flexibility of exchange, reuse, replacement, versioning, and tracing of the individual models. Third, not all analytics models, specifically defined by newly developed algorithms, are covered by the current PMML specification.

DMG is developing another standard entitled PFA (Portable Format for Analytics)[15] to fill this gap. Rather than using XML-based data schema, PFA proposes to use JSON (JavaScript Object Notation) based document to encapsulate workflow-based data processing procedures and provides support to sharing workflow states among different scoring engines. PFA shares the commonality with PMML that both are model specifications instead of implementations, whereas PFA's focus is on the scoring procedure. PFA uses the Data Pipeline concept to separate the data flow from the functions that are

[15] Portable Format for Analytics (PFA), http://dmg.org/pfa/

performed on data. PFA also offers a large suite of primitives that allow building different type of algorithms. As a result, new emerging models can be expressed by composing library functions or passing user-defined callbacks. This work would complement to PMML but it is still in the preliminary developmental stage, and it has not been widely supported by commercial data analytics tools (Li et al., 2015a).

Information Model for Data Analytics Over Product Lifecycle Stages

From the lifecycle perspective, many recent literatures have addressed issues related to sensor-data acquisition, data processing, analytics-model building and scoring, visualization and user interaction, as well as security and privacy (Assuncao et al., 2015). However, only a few literatures have presented works related to the management of the analytics models and their composition over time, which is needed to keep track of the changes to physical products as they move through their lifecycle.

One such work is CL_2M (Closed-Loop Lifecycle Management) (Kiritsis, 2011), which is an effort to extend PLM for smart products. It incorporates the PEID (Product Embedded Information Device) technology to collect a product's tag and sensor data, plus other necessary "lifecycle-event" data. As a result, CL_2M can provide feedback knowledge into the processes that make the product lifecycle. This work focuses primarily on the data management issues of product identification and sensor data. Another such work is reported by Jain et al. (2008). Specifically, the authors discuss the challenges regarding building, updating, and sharing complex data-mining models across the model lifecycle. However, their work focuses merely on analytics model management without considering their use for a product.

More recently, along with the Industry 4.0 concept for smart factory and smart manufacturing becoming influential, several German associations (BITCOM, VDMA, ZVEI/VDI/DKE, DIN) and international standardization organizations (IEC and ISO) proposed the reference architecture model for

Industry 4.0 (briefly called RAMI4.0). The RAMI4.0 architecture and its I4.0 Component (Industry 4.0 Component) implementation are aimed at modeling a smart factory that involves diverse asset objects related to the product, production, and services, in physical or virtual forms.

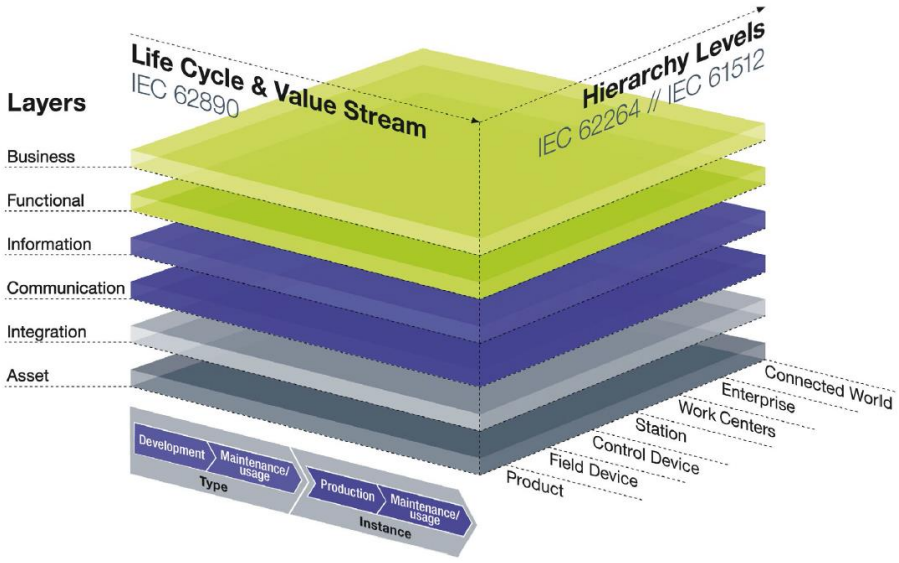


Figure 2.3 The Reference Architecture Model for Industry 4.0 (Adolphs et al., 2015)

The RAMI4.0 model expands the hierarchy levels of IEC/ISO 62264[16] (which is based on ANSI/ISA-95[17]) by adding a "Product" level and a "Connected World" level to extend the boundaries of an individual factory (Adolphs et al., 2015), as shown at the right horizontal axis in Figure 2.3. The left horizontal axis is used to represent the combination of *lifecycle* and *value stream* of a product or a system. It establishes the distinction between *Type* and *Instance* to support the representation of lifecycle and value stream. A type can be an activity from placing orders, development and testing, up to the first sample and prototype production. An instance is each manufactured product of a type. Improvements reported back to the manufacturer of a product from after-sales phases can lead to an amendment of the

[16] IEC 62264-1:2013 Enterprise-control system integration – Part 1: Models and terminology, http://www.iso.org/iso/catalogue_detail.htm?csnumber=57308
[17] ISA95, Enterprise-Control System Integration, https://www.isa.org/isa95/

type. Finally, the six layers (*Asset*, *Integration*, *Communication*, *Information*, *Functional*, and *Business*) on the vertical axis define the metadata of an Industry 4.0 component (I4.0 component for brevity) at the different abstraction level.

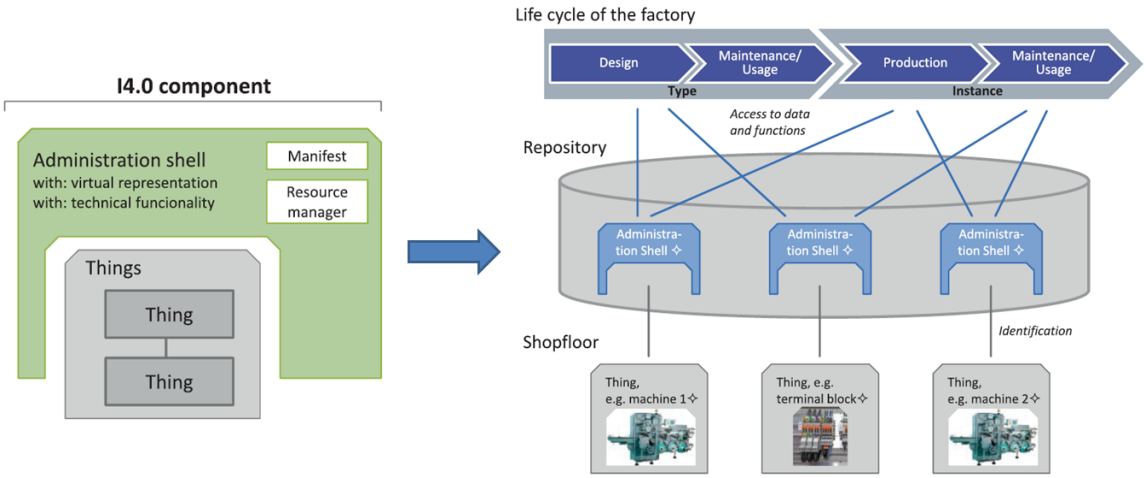


Figure 2.4 I4.0 Component and the Repository for I4.0 Components (Adolphs et al., 2015)

The *I4.0 Component* is derived from the Digital Factory Asset of IEC 62832[18]. I4.0 Component is a unified model for description of assets (from sensor/actuator till the whole plant), products, and all intellectual property (IP) used in a plant. An I4.0 component can be a production system, an individual machine or station, or an assembly inside a machine. It consists of an asset enriched by an *Administration Shell* that contains the virtual representation of the real asset, status data of the asset, and all data generated during the asset lifecycle. The administration shell is the central data warehouse for the asset during the whole lifecycle (Figure 2.4).

I4.0 Component is probably a candidate to model smart products because of the data and information container provided by the administration shell and the flexibility to implement it. However, how to abstract the virtual representation of different assets and how to implement the necessary

[18] IEC/TS 62832-1: Industrial-process measurement, control and automation Digital Factory framework Part 1: General principles, http://www.iec.ch/dyn/www/f?p=103:38:0::::FSP_ORG_ID,FSP_APEX_PAGE,FSP_LANG_ID,FSP_PROJECT:1250,23,25,IEC/TS 62832-1 Ed. 1.0

elements of the administration shell are out of the scope of the RAMI4.0 framework; they rely on the individual existing industrial or open standards and the practitioners who are implementing them.

2.1.2 Process Models for Transdisciplinary Collaboration

Regarding engineering processes, numerous process models have been proposed and adopted to understand, improve, and support the design and development processes for physical products as well as for software products. These process models either define the project structures at the *macro-level*, end-to-end flows of tasks at the *meso-level*, or individual process steps and their immediate contexts at the *micro-level* (Wynn and Clarkson, 2018). There is no means one individual model could cover all the necessary tasks and activities for a product development project. The practitioners have to select and adapt appropriate models for their needs. Since our target user roles are engineers and data scientists, we explore the New Product Development (NPD) process models for our baseline engineering process and the Knowledge Discovery and Data Mining (KDDM) models for the data science process. Below, we describe these models and then discuss how we think about incorporating KDDM into NPD.

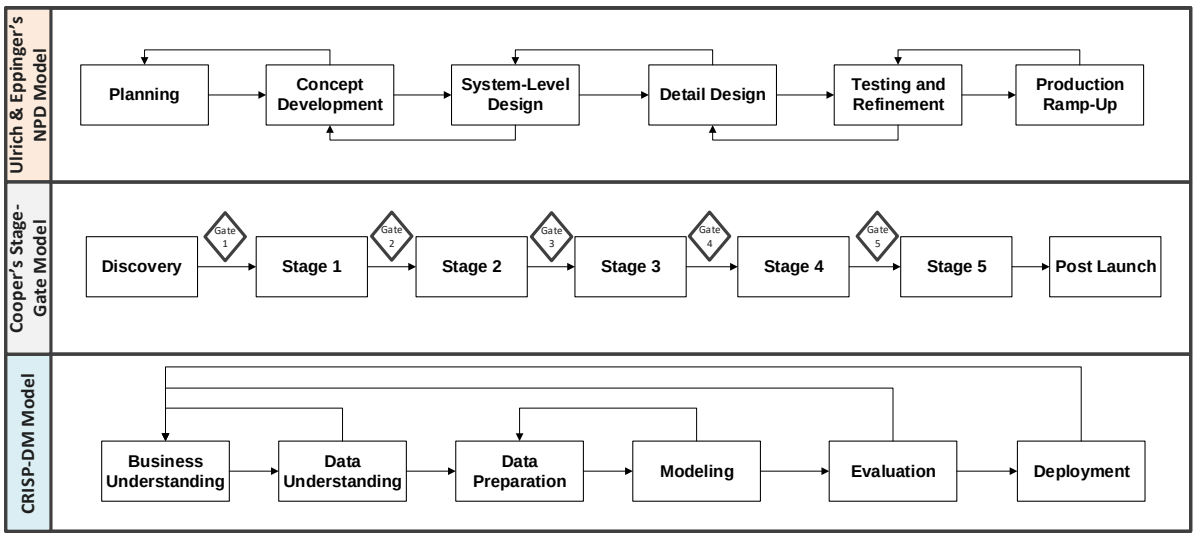


Figure 2.5 The Cooper's generic Stage-Gate model, the Ulrich & Eppinger's NPD model, and the CRISP-DM KDDM model

New Product Development (NPD)

New product development (NPD) transforms a market opportunity into a product (tangible or intangible) that is available to the market. There are two types of process models that are typically used in the traditional product development process (PDP). A sequential process model, also known as linear or waterfall, is a stage-gate-based process that has dominated in the manufacturing industry for several decades and is also often used for development of large-scale software systems.

Many NPD models have been adapted from the Cooper's Stage-Gate model (Cooper, 1994; 2008) that typically consists of a series of stages followed by gates (the middle lane of Figure 2.5). The prescribed stages and the criteria for transiting from one stage to the next provides useful guidelines to practitioners using the process. There are many versions of this process, such as the Ulrich and Eppinger's model, which is one of the well-adopted stage-gate models for physical product development. It consists of six high-level stages: *Planning*, *Concept Development*, *Sub-System Design*, *Detail Design, Testing and Refinement*, and *Production Ramp-up* (Ulrich and Eppinger, 2012), see the upper lane in Figure 2.5. Each stage is further prescribed with steps and activities.

An alternative process model is a spiral process that incorporates cross-phase iterations (Unger and Eppinger, 2009). The spiral process is commonly used in the software industry in the form of an Agile methodology. For example, an agile scrum methodology typically consists of a number of short development cycles (two to four weeks) undertaken by a dedicated project team.

The trend of mixing agile and stage-gate processes has been seen recently in manufacturing companies (Karlström and Runeson, 2005; Karlström and Runeson, 2006; Cooper, 2014; Cooper, 2016), particularly in high-tech companies developing large-scale mechatronics that consist of mechanical parts, electronic parts, and software (Eklund and Bosch, 2012; Eklund et al., 2014; Conforto and

Amaral, 2016). The Agile-Stage-Gate hybrid model combines the predictability and planning that is typically desired within manufacturing physical products with the dynamic capabilities of modern agile software development. This Agile-Stage-Gate model results in faster product releases, as well as better handling of changing customer needs, and improved team communication and morale (Cooper, 2016). However, there are some challenges causing manufacturers to be hindered in the adoption of agile practices. The primary difficulty is that the development of a physical product cannot be easily incrementalized, in that creating a potentially releasable, working product in a short-sprint is not usually feasible. Furthermore, developing a mechanical part often includes developing and investing in very expensive manufacturing tools with long lead times, which can expand the development cycle for the product to twelve months or longer.

For these reasons, agile methodologies are currently employed mainly in the development and testing phases of a product development project. The overall product development approach at an organization level is still governed by a stage-gate model (for project management) or single-cycle vee-model (for systems engineering).

Knowledge Discovery and Data Mining (KDDM)

As discussed previously, data analytics is indeed a production process for producing data products, taking data as materials, turning data into usable knowledge models, and delivering results based on data. Data products development (DPD) produces software-like but data-centered products (e.g. data processing pipelines, statistics and machine learning algorithms, and mathematical analytics models). The data analytics process is different from the traditional software development process because of the requirement to monitor and tune the model in short iterations and the fact that it is difficult for data scientists to know *a priori* what will be found when "exploring the data" (Saltz, 2015). However, similar to software development, the data analytics process is iterative by its nature, and data scientists require

constant revalidation of the problem, data sources, and outcomes.

Formal process models for data analytics projects originated from the knowledge discovery and data mining (KDDM) community. CRISP-DM (CRoss-Industry Standard Process for Data Mining) is one of the more successful process models that has been adopted by both industry and academia (Kurgan and Musilek, 2006). CRISP-DM is a waterfall model that prescribes six high-level phases (the lower lane in Figure 2.5) to formally describe a data analytics project and each phase is further decomposed into several key tasks and deliverables (Shearer, 2000). The CRISP-DM's six high-level phases – *Business Understanding*, *Data Understanding*, *Data Preparation*, *Modeling*, *Evaluation*, and *Deployment* – appropriately capture the necessary lifecycle stages for data science activities (Li et al., 2015b). This provides the possibilities to align the KDDM tasks with the NPD tasks to formulate a data-driven product development process, with appropriate adaptation and complementation of both models. Research has shown the two early stages, Data understanding and Data preparation, can take up to 65% of the overall data analytics efforts (see Table 2.2).

Table 2.2 Data Science lifecycle and responsibility assignment matrix (Sapp, 2017)

Task (Proportion of Effort)	Subtasks	Stakeholder: Business	Stakeholder: Data Scientist	Stakeholder: IT/ Operations
1. Problem Understanding (5% to 10%)	a) Determine Objective	x	x	
	b) Define Success Criteria	x	x	
	c) Assess Constraints	x	x	x
2. Data Understanding (10% to 25%)	a) Assess Data Situation	x	x	x
	b) Obtain Data (Access)		x	x
	c) Explore Data	x	x	x
3. Data Preparation (20% to 40%)	a) Filter Data		x	x
	b) Clean Data		x	x
	c) Feature Engineering	x	x	
4. Modeling (20% to 30%)	a) Select Model Approach		x	
	b) Build Models		x	
5. Evaluation of Results (5% to 10%)	a) Select Model		x	
	b) Validate Model		x	
	c) Explain Model	x	x	
6. Deployment (5% to 15%)	a) Deploy Model		x	x
	b) Monitor and Maintain	x	x	x
	c) Terminate	x	x	x

Incorporating KDDM in NPD

Most engineering design models prefer sequential models, focusing on an individual domain rather than considering interactions with other domains from a system perspective (Gericke and Blessing, 2011). Integrated Product Development (IPD), or Concurrent Engineering (CE), is an effective means to address overlaps and interactions between multidisciplinary activities in the new product development process, increasing the need to coordinate and be compensated through other aspects of the NPD process (e.g., integrated tools), product definitions (e.g., incremental development), organizational context (e.g., reduced task specialization), and teaming (e.g., cross-functional teams) (Gerwin and Barrowman, 2002). These traditional models have not explicitly addressed how to do data analytics during those processes. Systems Engineering, another effective approach for developing multidisciplinary, large-scale, complex systems recently introduced the data-centric perspective (Wheatcraft et al., 2017). The concept is focused on the formalized use of a common, integrated dataset to support concept maturation, requirements analysis, design, analysis, verification and validation activities. This integrated dataset represents the work product and the underlying data and information generated during each lifecycle phase of the product. Similar to IPD/CE, the systems engineering framework has not explicitly prescribed a data analytics process.

Incorporating a KDDM process into an NPD process presents two challenges. The first challenge is related to the current data analytics practices as conducted in manufacturing firms. The data varies significantly across a product's full lifecycle (Kassner et al., 2015). The product, production, and services related data is available in various manufacturing information systems (e.g., PLM, MES, and ERP systems) (Roy et al., 2014), but might also reside in an external supply chain partner's system. There can also be an inability of big data processing within manufacturing firms due to the limitations of IT resources within those firms (Sun et al., 2017). For example, few manufacturing experts are familiar

with modern big data analytics techniques. If the data analytics tasks that have previously been embedded in the engineering processes could be decoupled, it would be more efficient for those tasks to be done by a dedicated data science team. The question is, what are those embedded tasks?

The second challenge is related to the natural latency between the physical product development activities and the data products development activities. This is because the development of accurate analytics models greatly relies on new data generated as part of the physical product development process and there is an inevitable time lag between these development processes (Li et al., 2015b). In other words, while the NPD and KDDM processes both follow similar high-level stage sequences, there is no systematic way to synchronize the two sides' activities. Consequently, the different cycle times of physical product development and data products development can lead to less optimal solutions where issues are solved in software or data analytics (upgrade to the product) even though they would have been better solved in physical design (new generation of product), or vice versa. Understanding the interaction patterns of engineers and data scientists would be beneficial to the integrated decision-making process design, which in return facilitates a better system architecture design, for the development of a smart product.

2.1.3 Modular Product Architecture for Product Individualization

The ever-evolving nature of smart products need to incorporate changeability into the product architecture to enable flexibility, agility, adaptability, and robustness. Modularity (or Encapsulation) has been recognized as a key principle to realize these characteristics.

What is Modularity?

Modularity is an attribute of a system related to structure and functionality (Miller and Elgard, 1998). A *modular* structure is a structure consisting of self-contained, functional units, with standardized interfaces and interactions in accordance with a system definition.

Modularity originates from Suh's independence axiom: "in good design the independence of functional requirements is maintained" (Suh, 1990). Modularity has been proved to be important for mechatronic products because the short technology lifecycle of many of the functions in these products, combined with the customer demand for wide variety of features, necessitates designers to optimize the modularity of components and subassemblies for manufacturability and serviceability (Ishii, 1997).

While the debate on modularity versus integrity continues (Hölttä-Otto and de Weck, 2007), the concept of modularity has been well researched and accepted by the product design and manufacturing communities to balance the conflicts between product standardization and variation in the past. This concept has also been extended to operations management research to optimize service architecture design (Voss and Mikkola, 2009), supply chain design (Bask et al., 2009), and organizations (Langlois, 2000). It has been found that the use of modularity concept an efficient and effective means in realizing sufficient product variations to satisfy a wider range of customer demands, but the presence of data products (analytics models) and their frequent changes in smart products pose additional problems that need to be studied carefully.

Modular Product Architecture

Product architecture is the arrangement of the functional elements of a product into several physical building blocks, as well as specifying the interfaces among interacting physical components. For a smart product that integrates physical components and analytical components, the traditional definition of a product architecture may not hold true because the building blocks could be nonphysical. However, the

goal of a modular architecture is the same as per minimizing the physical changes required to achieve a functional change. Indeed, modular architecture design has been shown in building advanced data analytics and complex machine learning models, for example, Ensemble classification (Ponti, 2011; Asmita and Shukla, 2014). We will discuss this in details in Chapter 3.

A modular product architecture can be achieved by appropriate product family classification and product platform design. Two modular design approaches are often used to define product families (Simpson et al., 2005): (1) the *Module-based* approach that allows product family members being instantiated by adding, substituting, and/or removing; and (2) the *Scale-based* approach that allows one or more scaling variables to be used to stretch or shrink the product in one or more dimensions. Figure 2.6 shows two product families that are designed by these two approaches, respectively.

Figure 2.6 (a) Module-based example: Caterpillar excavator[19]; (b) Scale-based example: Dewalt wrench tools[20]

Modular design clusters the system's functions into various modules while minimizing the coupling among the modules and maximizing the cohesion within the modules. Modular design thus has been a key to mass customization to reduce internal product variety but maintain the variety of customization.

[19] Caterpillar Excavator, www.cat.com
[20] Dewalt Wrench Tools, www.dewalt.com

Modular product architectures to construct product families and platforms are usually developed to fulfill the mass customization process.

Modularity Quantification of Product Architecture

A number of quantitative methods to measure the degree of modularity have been previously reported. Each method has its own emphasis: either measuring component similarities, examining the variations in component sharing, or estimating the impact of design alternatives. For example, Gershenson et al. (1999) proposed a measure of relative modularity: $M = S_{in}/(S_{in} + S_{out}) + D_{in}/(D_{in} + D_{out})$, where S_{in} measures the similarity between components within a module, S_{out} measures the similarity between a component within a module and other components outside of the module. Similarly, D_{in} measures the dependency between components within the module, and D_{out} measures the dependency between a component within a module and a component outside of the module. This measure does not differentiate the standard components and nonstandard components in the module.

One attempt to integrate different factors is Mikkola's modularity function, *M*(*u*) (Mikkola and Gassmann, 2003). It is defined as a function of the unique components (*u*) in a given product architecture. It consists of another three variables to reflect the inherent characteristics that contribute to modularity: the total number of components (*N*), the substitutability factor (*s*), and the degree of coupling (δ): $M(u) = e^{-u^2/2Ns\delta}$, where

- *N*: the total number of components. A component is defined as a physically distinct portion of the product that embodies a core design concept and performs well-defined function.
- *u*: the number of NTF (New-To-Firm) components. NTF components refer to product-specific components that are introduced to the manufacturing firm for the first time.
- *n* = *N*-*u*: the number of standard components. Standard components refer to components that have been used in previous or existing architectural designs by the firm.
- *k*: the number of interfaces of an NTF component. Interfaces are linkages shared among components.

- δ: the degree of coupling. It is defined as the ratio of the number of interfaces (k) by the number of components (N) in a subsystem of a given product architecture.
- s: the substitutability factor. It is estimated as the number of product families made possible by the average number of interfaces of NTF components required for functionality.

This function has been modified for assessing the degree of modularity for mass customization product architecture (Mikkola, 2007), and the degree of modularity for service architecture (Voss and Mikkola, 2009).

CSP Problem Formulation and Solving Methods

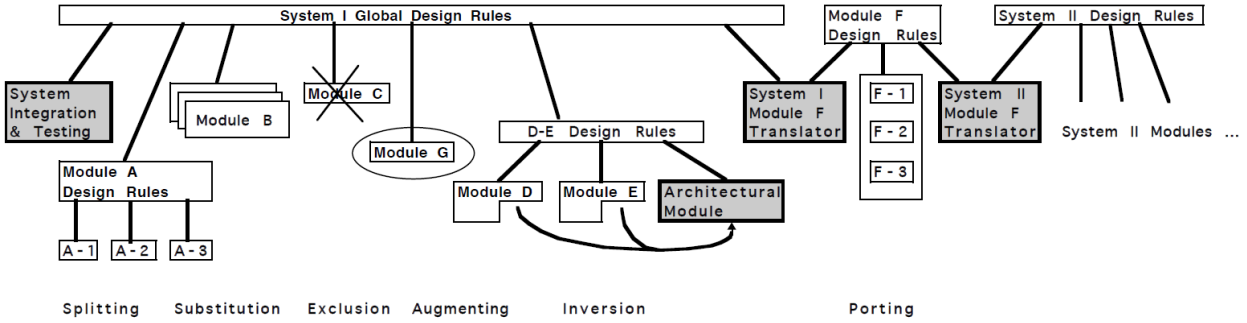


Figure 2.7 The effect of the six operators on a modular system (Baldwin and Clark, 2006)

There are six core operators to manipulate modules (Baldwin and Clark, 2006) in order to design a modular system. Their effects are illustrated in Figure 2.7:

- *Splitting*: modules can be made independent.
- *Augmenting*: new modules can be added to create new solutions.
- *Excluding*: existing modules can be removed to build a usable solution.
- *Substituting*: modules can be substituted and interchanged.
- *Inverting*: the hierarchical dependencies between modules can be rearranged.
- *Porting*: modules can be applied to different contexts.

This enables the product configuration based on component modules. Configuration is a generative process, and a solution is the result of a search process, within the space of all possible combinations of

objects (Sabin and Weigel, 1998). A generic configuration task can be defined as a constraint satisfaction problem (CSP). A literal definition was given by Mittal and Frayman (1989):

Given: *(a) A fixed, pre-defined set of components, where a component is described by a set of properties, ports for connecting it to other components, constraints at each port that describe the components that can be connected at that port, and other structural constraints; (b) some description of the desired configuration; and (c) possibly some criteria for making optimal selections.*

Build: *One or more configurations that satisfy all the requirements, where a configuration is a set of components and a description of the connections between the components in the set, or, detect inconsistencies in the requirements.*

A Static CSP problem assumes the constraints are not changed (Subbarayan, 2005) and it can be mathematically formulated as below:

1. *Let X be a set of variables* $\{x_1, x_2, \cdots, x_n\}$ *and D be the set* $\{D_1, D_2, \cdots, D_n\}$*, where* D_i *is the domain of values for variable* x_i.
2. *A relation R over the variables in M,* $M \subseteq X$*, is a set of allowed combinations of values for the variables in M. Let* $M = \{x_{m1}, x_{m2}, \cdots, x_{mk}\}$*, then* $R \subseteq (D_{m1} \times D_{m2} \times \cdots \times D_{mk})$*. R restricts the ways in which the variables in M could be assigned values.*
3. *A constraint satisfaction problem instance CSP is a triplet (X, D, C), where* $C = \{c_1, c_2, \cdots, c_m\}$ *is a set of constraints. Each constraint,* c_i*, is a pair* (S_i, R_i)*, where* $S_i \subseteq X$ *is the scope of the constraint and* R_i *is a relation over the variables in* S_i.
4. *An assignment is a pair* (x_i, v)*, where* $x_i \subseteq X$ *and* $v \subseteq D_i$*. The assignment* (x_i, v) *bounds the value of* x_i *to v. A partial assignment, PA, is a set of assignments for all the variables in Y, where* $Y \subseteq X$*. A partial assignment is complete when* $Y = X$.
5. *Let* $var(PA) = \{x | (x, v) \in PA\}$*, the set of variables assigned values by a PA. Let* $val(PA) = \{v | (x, v) \in PA\}$*, the set of values assigned for var(PA). A partial assignment PA satisfies a constraint* c_i*, when* $PA_{i|var(PA) \cap S_i} \in R_{i|var(PA) \cap S_i}$.
6. *A complete assignment CA is a solution S for the CSP when CA satisfies all the constraints in C.*

If the requirements, constraints, and components dependencies are frequently changed, which are common in more realistic problem, a Dynamic CSP (DCSP) problem formulation is more appropriate. A DCSP can be seen as a sequence of static CSPs each resulting from the addition or retraction of a

constraint in the preceding one.

The classical methods to solve CSP problems include: *rule-based reasoning*, *model-based reasoning*, and *case-based reasoning* (Sabin and Weigel, 1998). A rule-based system, also known as expert system, uses production rules for representing both domain knowledge and control strategy. The lack of separation between domain knowledge and control strategy makes the knowledge-maintenance task difficult, extremely in large rule-based systems (Li et al., 2015a). Model-based reasoning decomposes entities and interactions between their elements. It separates between what is known and how the knowledge is used, thus enhances reusability of existing knowledge and compositionality of knowledge from different domains within a single model. Case-based reasoning solves the configuration problem by finding a similar, previously solved problem and adapting it to the new requirement. It does not need *a priori* complete model in the system; rather, it can always propose a solution to the user if it can find a similar case in the case base. If no complete model is available, it can use model-based methods to modify the mostly similar configuration.

In all these configuration systems, knowledge elicitation, representation, and new knowledge incorporation are the key for effectiveness and efficiency of generating optimal solutions. It is needed to have a unified modeling method to generalize the compiled, explicit knowledge and dynamic, implicit knowledge, one knowledge classification suggested by Chandrasegaran et al. (2013). Literatures have reported that extracting knowledge explicitly from data mining or machine learning models into rule-based systems. For example, Deb et al. (2011) presented an integrated approach that uses the back-propagation ANN (Artificial Neural Networks) methodology to construct prior domain knowledge and uses a CLIPS (C Language Integrated Production System) rule-based expert system to automate the tasks of setup formation, operation sequencing and datum selection for rotationally symmetrical parts for manufacturing process planning. In a previous work, we also investigated the issues of translating facts

and rules in expert systems into appropriate predictive models (Li et al., 2015a). This work had shown the possibility of using open standards (e.g. PMML) to generalize the top-down expert-driven rule models and the bottom-up data-driven predictive models.

2.1.4 Product/Process Modeling and Analysis Methods

Design Structure Matrix

Design Structure Matrix (DSM) is a methodology for modeling complex systems. It is designed for systems engineering of products, processes, and organizations, specifically for system decomposition and integration (Browning, 2001). A DSM is a matrix representation of a directed graph that represents a complex system. It uses a square matrix (NxN) with identical row and column labels. Elements are represented by the shaded cells along the diagonal; an off-diagonal mark indicates the dependency of one element on another. Figure 2.8 shows the DSM representation of the CRISP-DM process model shown previously (Figure 2.5). In this activity-based DSM, the Data Preparation activity precedes the Modeling activity; and, the Modeling activity goes back to the Data Preparation activity in case of a new iteration. Accordingly, the super-diagonal matrix cells show feedforward information, while the sub-diagonal cells indicate feedbacks (the iterations).

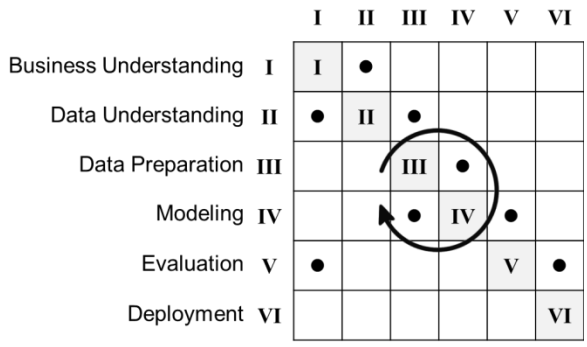


Figure 2.8 The DSM representation of the CRISP-DM process

There are two main categories of DSMs: *static* and *time-based*. Each category contains two types of

DSMs. Static DSMs represent system elements existing simultaneously; they include Component-based DSM and Team-based (or organizational) DSM. In time-based DSMs, the ordering of the rows and columns indicates a flow through time: upstream activities in a process precede downstream activities. Activity-based DSM and Parameter-based DSM are time-based DSMs.

Table 2.3 DSM classification (Browning, 2001; Yassine, 2004; Bartolomei et al., 2007)

DSM Category	*DSM Data Type*	*Representation*	*Application*	*Analysis Method*
Static DSMs	Team-based DSM	Multi-team interface characteristics	Organizational design, interface management, team integration	Clustering
	Component-based DSM	Multi-component relationships	System architecting, engineering and design	Clustering
Time-based DSMs	Activity-based DSM	Task/Activity input/output relationships	Project scheduling, Activity sequencing, Cycle time reduction	Partitioning, Tearing, Banding
	Parameter-based DSM	Parameter decision points and necessary precedents	Low level activity sequencing and process construction	Partitioning, Tearing, Banding

Table 2.4 Analytical methods for DSM (Browning, 2001; Yassine, 2004; Bartolomei et al., 2007)

Analytical Method	*Task*	*Goal*
Partitioning/Sequencing	Reorder system components with time-based dependencies, to produce a lower or upper triangular matrix	• Maximize the feed-forward information flow by moving dependency marks closer to the matrix diagonal; • Minimize feedback to reduce rework; • Recognize necessary feedbacks
Clustering	Reorder matrix rows and columns by grouping highly related nodes	• Maximizing interactions inside a cluster; • Minimizing interactions between clusters; • Recognize necessary overlay between clusters that represent necessary information sharing; • Recognize subclusters within a major cluster
Banding	Add light or dark shading to a DSM to show independent nodes of groups of nodes	• Identify activities/components that are independent of one another so that can be treated simultaneously • Recognize the critical path to the project, where one node in each band is a potential bottleneck
Tearing	Identify various feedback marks that if removed from the matrix, and the matrix is repartitioned, will obtain a lower or upper triangular matrix	• Minimize number of tears; • Confine tears to the smallest blocks on diagonal

A variety of matrix-based analytical techniques were available to analyze information presented in DSMs (Browning, 2001; Yassine, 2004; Bartolomei et al., 2007). These techniques include: *Partitioning/Sequencing*, *Clustering*, *Banding*, and *Tearing* (see Table 2.3 and Table 2.4). Static DSMs

are usually analyzed with clustering algorithms, and time-based DSMs are typically analyzed using partitioning/sequencing algorithms.

Other analytical methods include *Sensitivity Analysis* and *Network Analysis*. Kalligeros (2006) presented a Sensitivity DSM technique to identify system elements that are sensitive to change. The technique uses a DSM containing uncertainty information of the impact due to the occurrence of a change event. Applying this technique on an off-shore oil drilling platform, he investigated whether a 100% change in one subsystem would cause at least 20% change in the related subsystem. Bartolomei (2007) presented a Network-based DSM to analyze interaction between stakeholders in the social domain. A variety of network metrics – *Betweeness*, *Path length*, and *Centrality* – generated by the social network community are used. Betweeness is a measure of the number of times a vertex occurs on a geodesic (the short path connecting two vertices); Centrality is a measure of the connectedness of each node; and Path length refers to the distance between pairs of nodes in the network. Bartolomei (2007) showed betweeness of the stakeholders is related to their influence on a social network, and degree of centrality is associated with power or importance. This can help identifying key system elements (stakeholders, components, and activities) that are highly connected within the system thus recognizing critical nodes to be carefully monitored (Bartolomei et al., 2007).

DSM has also been used in change propagation models for analyzing and visualizing which parts of the system are affected by change (Pasqual and Weck, 2012). A two-layer model is usually used to capture the propagation path based on the interdependencies among components: (1) a Delta-DSM captures the differences between a baseline system and a change system; and (2) a Change-DSM captures the change propagation paths between components of the system by showing the initiating components as one dimension and the receiving ones as another dimension. Then, quantitative measures are applied to calculate the propagation impact.

Domain Mapping Matrix

The square matrix representation restricts DSM for inter-domain analysis. It assumes two domains should contain an equal number of the same elements. Domain Mapping Matrix (DMM) extends the classical DSM to a rectangular matrix (MxN) to model interactions across domains (Danilovic and Browning, 2007). DMM generalizes the product development processes into five domains (see Figure 2.9): (1) the *Product* (or service, result) system; (2) the *Process* system; (3) the *Organization* system grouping people into departments, teams, and other smaller units; (4) the system of *Tools* and equipment; and (5) the system of *Goals* that are related to objectives, requirements, and constraints. Each domain can be further decomposed. Most analytical methods (e.g. clustering and sequencing) for DSM presented before can be applied to DMM. The only difference is the analytical dimensions and that clustering is no longer concentrating along the matrix diagonal.

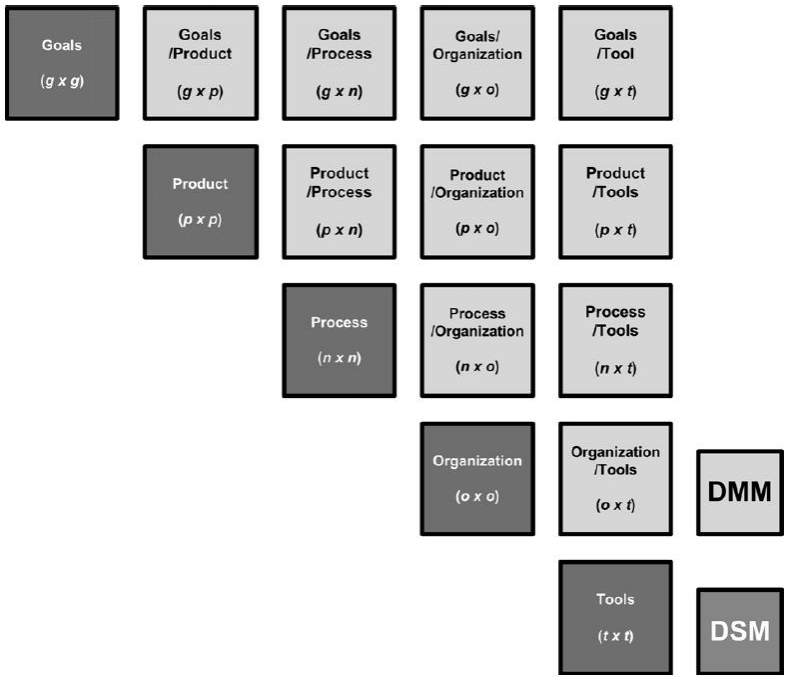


Figure 2.9 Domain Mapping Matrix (DMM) for five systems/domains (Danilovic and Browning, 2007)

The largest challenge of the DSM and DMM methods are their flat matrix representation

(Bartolomei et al. 2007; Danilovic and Browning, 2007). They do not allow clear representation of multiple types of relationships or node attributes. For example, it is difficult to represent multiple types of interfaces in one matrix cell when we develop Component-DSMs. Practitioners have to develop different techniques to overcome this restriction. For example, Helmer et al. (2008) argued that both *signal* type and *structural* type interactions are required to capture the mechatronics architecture and they proposed using Perspective Reduction (PR) and Data Reconciliation (DR) techniques to aggregate multi-interface values into one single value to represent the interface dependency between any two components within the product (Figure 2.10).

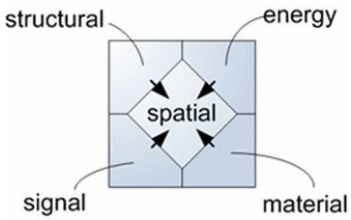


Figure 2.10 Using spatial type interface to aggregate the other four interface types (Helmer et al., 2008)

2.2 The Concept of "Smart Products Hypercube"

2.2.1 Physical-Analytical Dependency of Nest Thermostat's Components

Let's recall the Nest Thermostat's component decomposition matrix shown in Figure 1.14. That matrix was generated using the DSM technique discussed in previous section. The interfaces among any components within a product are usually classified into five types: *Spatial*, *Structural*, *Energy*, *Material*, and *Signal* (Sosa et al., 2003). Spatial dependency indicates that physical adjacency is needed for alignment, orientation, serviceability, assembly, or weight. Structural dependency indicates the existence of a functional requirement for transferring design loads, forces or containment. Energy dependency indicates a functional requirement related to transferring heat energy, vibration energy, electrical energy, or noise. Material dependency indicates a functional requirement related to transferring air, oil, fuel, or

water. Finally, signal dependency indicates a functional requirement related to transferring signals or controls.

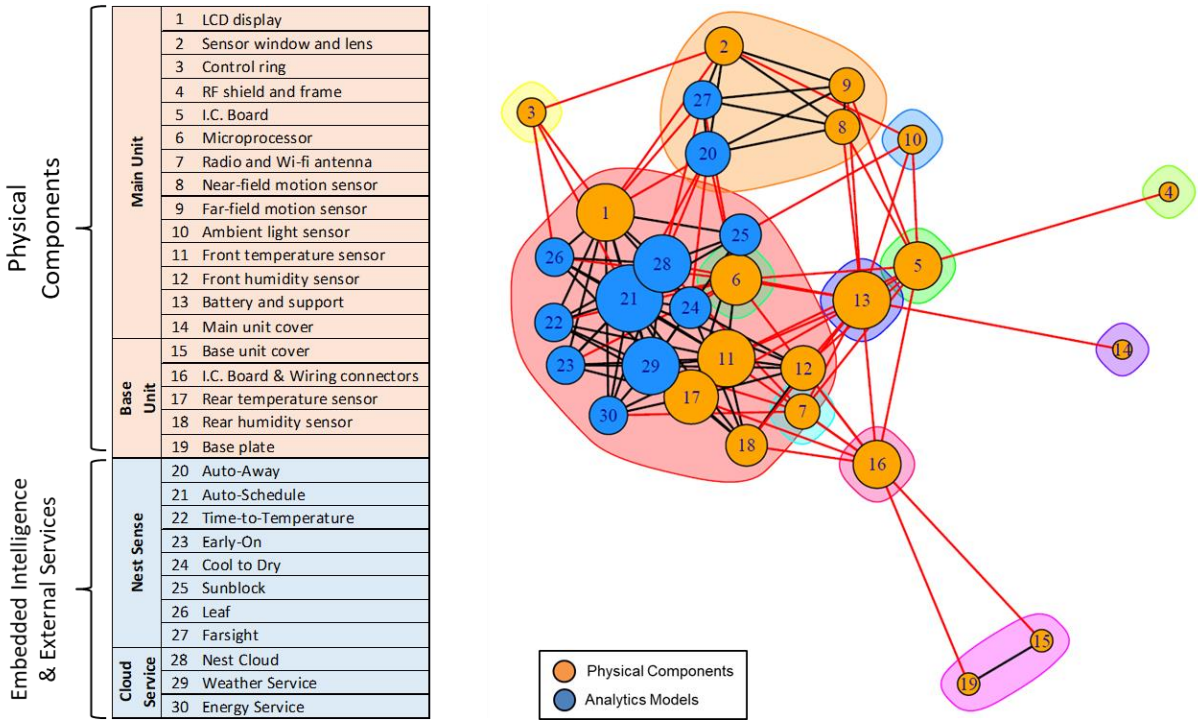


Figure 2.11 Nest Thermostat's component network

In order to understand the correlations between those physical components and the smart analytics features within the thermostat, we defined a connection between any two components if there are at least one of the five interface dependencies. The DSM matrix can be analyzed using network analysis techniques. After applying a *Graph Community Detection* algorithm (Newman and Girvan, 2003, refer to Appendix A for details), we obtained a network graph shown as in Figure 2.11. While many physical components and analytics features form one big, central community due to the integral nature of micro-controller-based mechatronics. We also observed that (1) those communities far from the central community contain the physical components mainly serving fixing or enclosing functions, for example, the base plate, covers, and RF shield; (2) physical components and analytics features can altogether form a "seemingly independent" community although it has connections to the components in other

communities. For instance, the field motion sensors (component No. 8 and 9) and the Auto-Away (No. 20) and Farsight (No. 27) features belong to one community, implying this collection of physical components and analytics features could be decoupled from the overall product architecture; they could be modularized even be standardized.

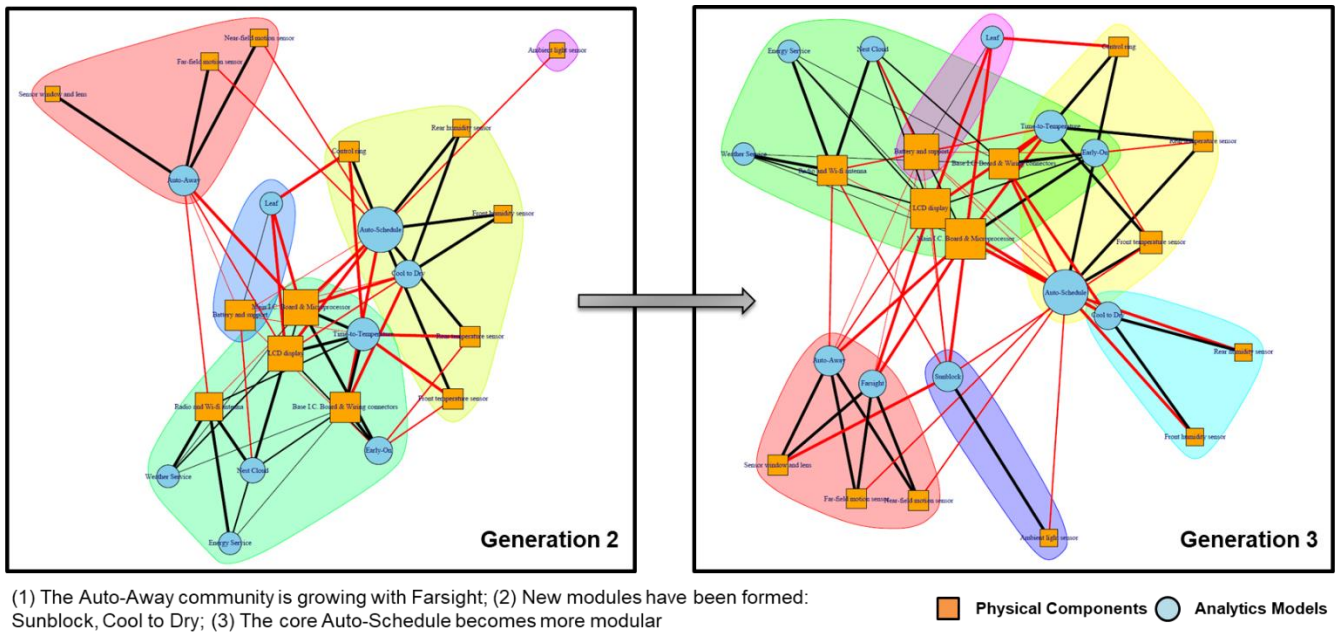


Figure 2.12 Modules formulated from physical components and analytics models in two generations of Nest Thermostat

Now, let's focus on the physical components and analytics features that have signal and/or spatial dependencies. Their interfaces can be modeled as a DMM. We developed two DMMs for the Generation 2 and 3 of the Nest Thermostat so as to observe its evolution pattern (see Appendix A for details). Two new graphs were then generated and are shown as in Figure 2.12. We observed how new communities were born and how the old communities did change. For example, the Auto-Away community is growing with a new analytics feature, Farsight, joining into the community; the two analytics features share the same sensors. Furthermore, two new communities are formed: the ambient sensor is now also used to develop a new analytics feature, the Sunblock; the correlations between the Cool to Dry feature and the humidity sensors become relatively stronger, separating them from the previous Auto-Schedule

community and forming a new community.

2.2.2 Smart Products Hypercube

Let's assume a smart product in general can be decomposed into two distinct components: (1) the *Physical Body* that is formed by *Physical Components* (mechanical parts, electrical parts, network components, sensors, etc.); and (2) the *Intelligence* that is implemented by various levels of *Analytics Models* (first-principle-based analytical model, numerical model, empirical model, machine learning model). Furthermore, both types of components can have lower-level children components and can be part of upper-level components. That is, a physical component could be a physical part, a sub assembly, or a sub system; An analytics model could be a unit model or a model ensemble.

The design of a smart product system requires a continuous evolution of its architecture by upgrading the product already in service, instead of always releasing a new version or derivative. Engineering change is therefore inevitable, and it is necessary to incorporate a changeability into the product architecture. The tight information dependency between an analytics model and its relevant physical components (e.g. sensors) implies that they should be treated as a component module to increase the modularity of the smart product. For the Nest Thermostat's Auto-Away feature, it relies on field-motion sensors to detect the user presence. The data generated from these sensors can also be used to develop another feature, the Farsight. This does not require changing other physical designs except a bigger screen for better user experience. In this case, the field-motion sensors and the analytics models to detect and predict user's location can be an atomic component, which can be reused by the two features.

Definition 1: *Smart Component***.** We define a *Smart Component (sComponent)* as a unit system that consists of the necessary physical components (for its form) and analytics model components (for its intelligence implementation) to achieve a reused function. Mathematically, a smart component can be represented as below,

$$sComponent = f(PCs, AMs, R_{PC,AM}, t) \tag{2.1}$$

where *PCs* are the necessary physical components, *AMs* are the necessary analytics models, $R_{PC,AM}$ are the dependencies between the physical components and the analytics models, and *t* indicates the time variable at which all the components information and relationships hold.

Definition 2: *Smart Product***.** A *Smart Product (sProduct)* is the composition of modular smart components. It is composed of one or more *Smart Components (sComponent)*, and has *Configuration Rules* to reflect their relationships (*Conf(sComponents)*):

$$SP = < sComponents, Conf(sComponents) > \quad (2.2)$$

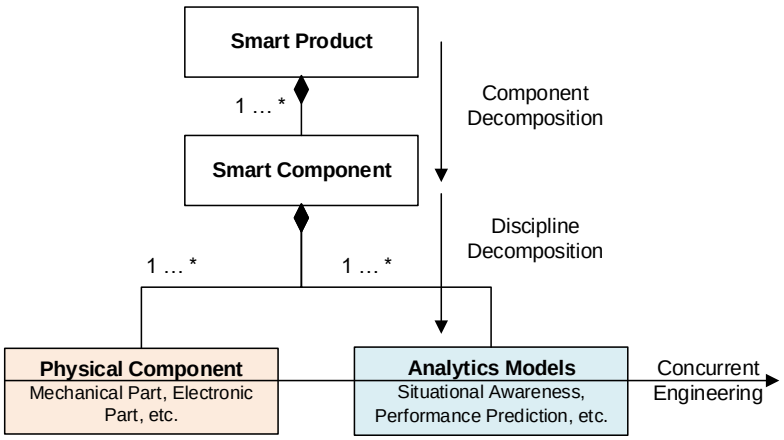


Figure 2.13 Smart Component and Smart Product

The two definitions are graphically shown in Figure 2.13. It has the benefit to enable concurrent engineering regarding disciplines being applied at the level of primitive components, and this provides the possibility for optimal designs. It is noted a smart component is a superset of physical components and analytics models, it can also be used to represent physical components or analytics models, if we relax the cardinality constraints (one to many) to allow zero component. Furthermore, the role of an analytics model as a part or a service (Section 1.2.4) in a product system can be determined by the spatial relationship, i.e. whether it is embedded inside the product or it runs on an external cloud.

The DSM and DMM techniques can be used to decompose and analyze the relationship between physical components and analytics models. However, in order to capture the information for a smart

product over its entire lifecycle: beginning-of-life (BOL), middle-of-life (MOL), and end-of-life (EOL), we have to introduce a high-dimensional representation model, namely, Smart Products Hypercube (sPH). Let's define the *Physical Body* as dimension *P*, the *Intelligence* as dimension *A*, and the product *Lifecycle* as dimension *T* (to represent the life evolution of the smart product). As shown in Figure 2.14, each cell within the hypercube then represents a smart component (or a smart product) with certain physical forms, certain level of intelligence, and at a certain life stage. An important feature is that any cell in this space contains incremental information of all its lower dimensional cells. This is consistent with the fact that a product evolution is often cumulative. Therefore, the rightest outer cell represents a system of smart products. Accordingly, we name the space defined by this hypercube as "P-A-T Space."

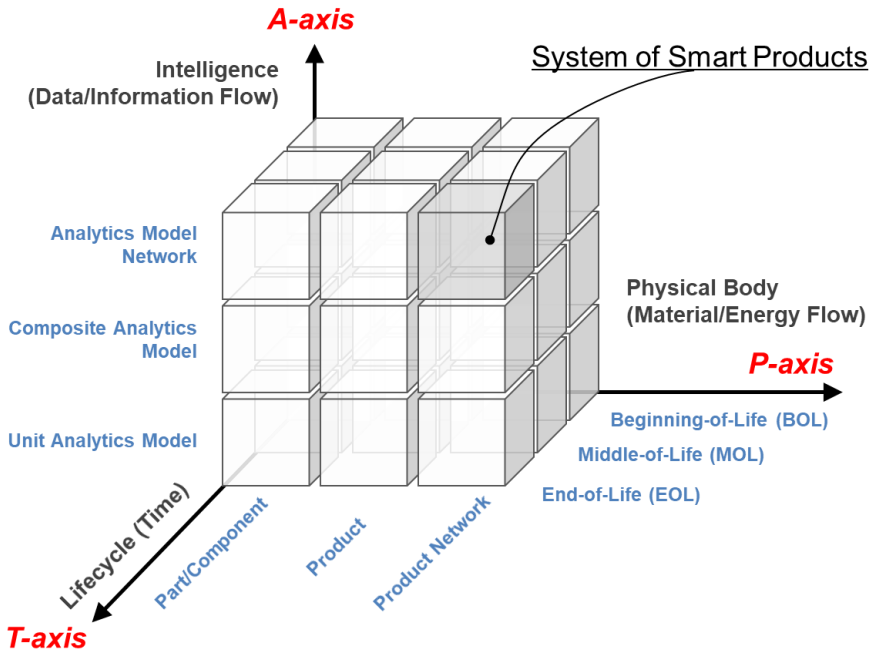


Figure 2.14 The Smart Product Hypercube (sPH) and its P-A-T Space

2.2.3 Problem Domain Decomposition

Now, let's look at the design space of the Smart Products Hypercube. The *hypercube* is a geometric representation and generalization of a *3-cube* to *n* dimensions, also called an *n-cube*. Mathematically, a

tensor is often used to characterize data in high dimensions using a multidimensional array. A first-order tensor is a *vector*, a second-order tensor is a *matrix*, and tensors of order three or higher are called higher-order *tensors*.

Increasing computing capacity and better understanding of multilinear algebra during the last decades have expanded the tensor concept and its analysis techniques into many domains such as statistics, data science, and machine learning (Kolda and Bader, 2009; Rabanser et al., 2017). Below, we use the tensor concept and notation to formally decompose the Smart Products Hypercube space. The detailed review of the tensor notations, definitions, and decomposition methods can be referred to Kolda and Bader (2009).

It is noted that our intention of using the tensor method is to rigorously identify and define our research spaces and domains. We are not using the tensor decomposition to carry out product design or analysis in the following chapters, but the author believes the discussion below would shed a light on fully exploring the tensor techniques for smart products development in future research.

Definition 3: *sPH Tensor*. We define the whole *P-A-T* space as an *sPH Tensor*, $\mathcal{H}$, to study the product and process information and relationships that are involved in the development of smart products over their lifecycle. The *Rank-one sPH Tensor* is defined as below and graphically it is shown as in Figure 2.15:

$$\mathcal{H} = p \circ a \circ t, h_{ijk} = p_i a_j t_k \tag{2.3}$$

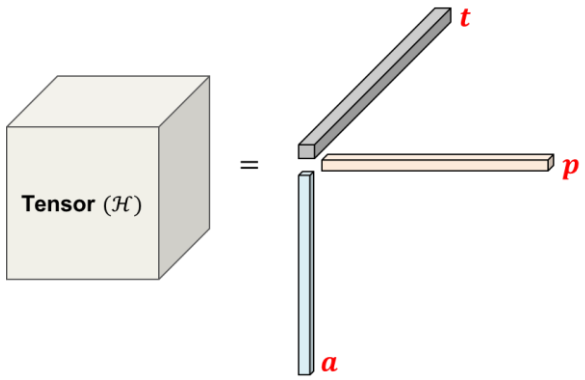


Figure 2.15 Rank-one sPH Tensor, $\mathcal{H} = p \circ a \circ t, h_{ijk} = p_i a_j t_k$

sPH Fibers and Slices

Definition 4: *sPH Fiber*. By fixing every index but one, a tensor can be decomposed into *Mode-1 (columns)*, *Mode-2 (rows)*, and *Mode-3 (tubes) Fibers*. In the smart products context, Mode-1 fiber, $h_{i:k}$, stands for a certain physical component with different level of intelligence; Mode-2 fiber, $h_{:jk}$, stands for a certain analytics model which can be used for different complexity of physical components; Mode-3 fiber, $h_{ij:}$, stands for a certain smart component at its entire lifecycle. Graphically, they are shown in Figure 2.16.

Example. If we look at one temperature sensor in the Nest Thermostat, the analytics model dimension for this sensor can be temperature data, the time-series prediction based on the historical temperature data, and the time-to-temperature feature that consumes the measured temperature data from this sensor. This is a Mode-1 fiber.
If we look at the GPS positioning function, it is used by the user's mobile phone to detect the user's location. This positioning function can also be used by the Nest Thermostat to enhance its auto-away prediction that also uses the thermostat's own near-field and far-field motion sensors to detect the human actions. This is a Mode-2 fiber.
The entire Nest Thermostat is a Mode-3 fiber. Any Nest Thermostat's independent module that has both physical components and analytics models is also a Mode-3 fiber.

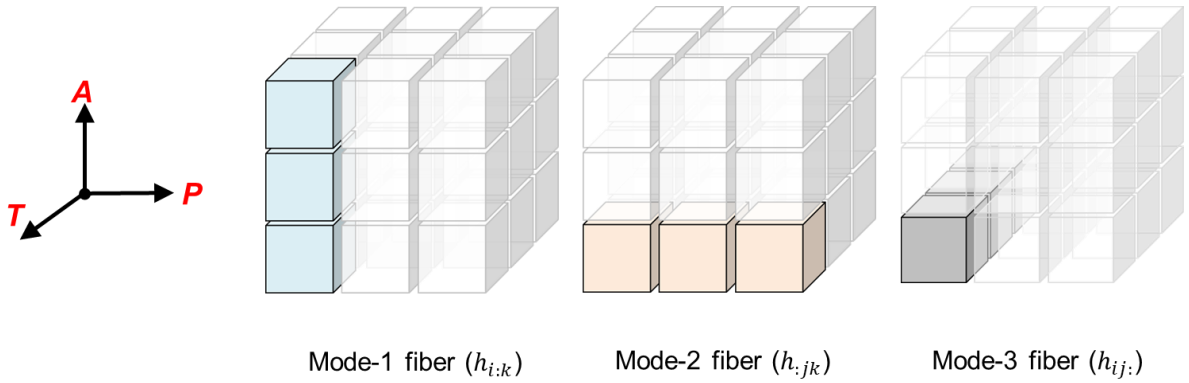


Figure 2.16 sPH Mode-1, Mode-2, and Mode-3 Fibers

Definition 5: *sPH Slice*. By fixing all but two indices, a tensor can be decomposed into *horizontal*, *lateral*, and *frontal* slices. In the smart products context, the horizontal slices or P-T planes, $H_{:j:}$, are related to the traditional PLM domain; the lateral slices or A-T planes, $H_{i::}$, are related to analytics model lifecycle management, a relative new domain; the frontal slices or P-A planes, $H_{::k}$, are related to how physical products interact with analytics models at any given decision points. Graphically, they are shown in Figure 2.17.

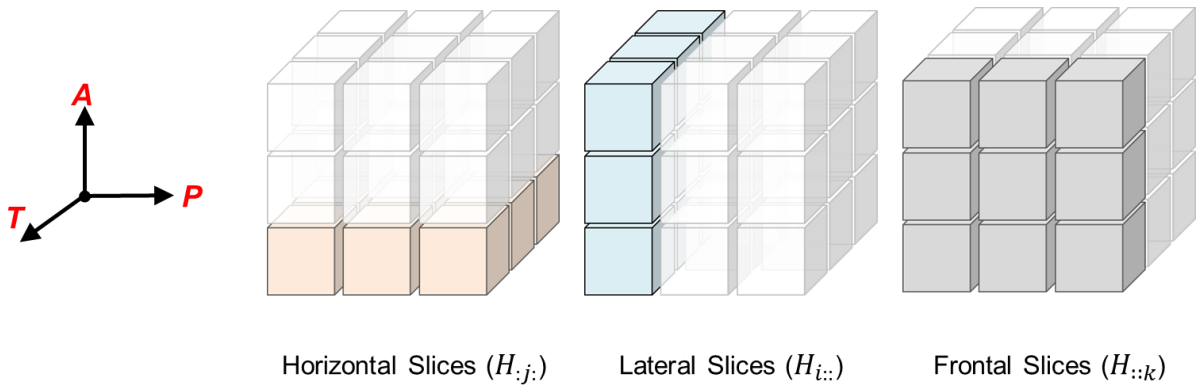


Figure 2.17 sPH Horizontal, Lateral, and Frontal Slices

The sPH design space is then decomposed into seven domains:

1) *Mode-1 fibers (A axis),* $h_{i:k}$: Hierarchical structure of *Analytics Models* for a specified physical component;
2) *Mode-2 fibers (P axis),* $h_{:jk}$: Hierarchical structure of *Physical Components* for a specified analytics model;
3) *Mode-3 fibers (T axis),* $h_{::k}$: Time-dynamic change of either physical product, analytics models, or their composition;
4) *Horizontal slices (P-T planes),* $H_{:j:}$,: related to the traditional *Product Lifecycle Management (PLM)* domain;
5) *Lateral slices (A-T planes),* $H_{i::}$: related to *Analytics Model Lifecycle Management*, a relatively new domain;
6) *Frontal slices (P-A planes),* $H_{::k}$: related to the interfaces or dependencies between physical products and analytics models, i.e. *Product Configuration*, at any given decision points; and
7) *Tensor (P-A-T space),* $\mathcal{H}$: related to all the necessary data and activity information about how a smart product evolves along its entire lifecycle (from conception to design to disposal), we name it *Smart Products Lifecycle Management (sPLM)*.

sPH Decomposition

Similarly, we can define sPH decompositions using the concepts defined in the tensor decomposition. Here, we present two types of decompositions for the smart products hypercube and explain their implications in smart products development.

Definition 6: *sPH CP Decomposition.* This is the canonical polyadic (CP) decomposition (also known as PARAFAC or CANDECOMP) that expresses a tensor as the sum of a finite number of rank-one tensors. We define the *sPH CP Decomposition* as

$$\mathcal{H} \approx \sum_{r=1}^{R} p_r \circ a_r \circ t_r \tag{2.4}$$

Where $p_r \in \mathbb{R}^I$, $a_r \in \mathbb{R}^J$, $t_r \in \mathbb{R}^K$, and their three-way outer product[21] is given by

$$(p_r \circ a_r \circ t_r)(i,j,k) = p_r(i)a_r(j)t_r(k) \quad \text{for all } i, j, k.$$

The *factor matrices* are defined as

$$P = [p_1\ p_2\ \cdots\ p_R] \in \mathbb{R}^{I \times R}$$
$$A = [a_1\ a_2\ \cdots\ a_R] \in \mathbb{R}^{J \times R}$$
$$T = [t_1\ t_2\ \cdots\ t_R] \in \mathbb{R}^{K \times R}$$

sPH CP decomposition can be used to describe the module-based product architecture design. The individual variables and vectors can represent the metadata of physical components, analytics models, and their time states. The factor matrices can be used to describe the interfaces, dependencies, and configuration rules in each configuration. Graphically, it is shown in Figure 2.18.

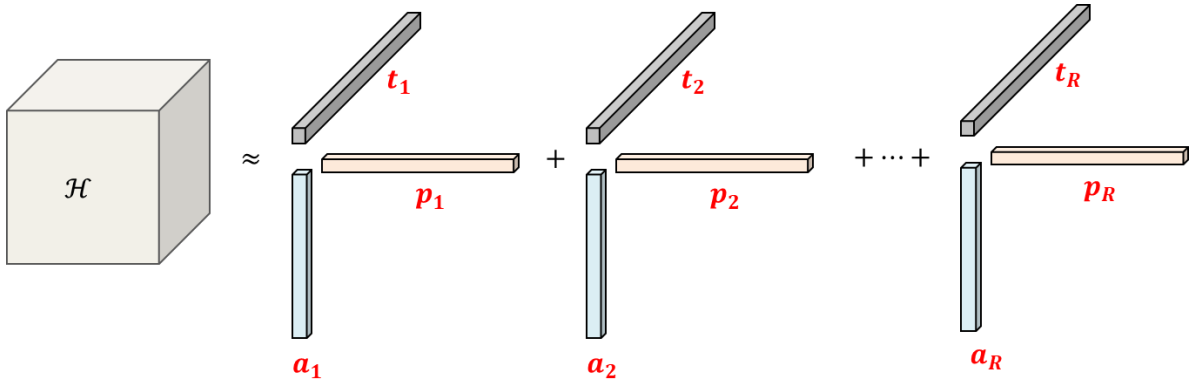


Figure 2.18 sPH CP decomposition

Example. As shown previously (Figure 1.12, Chapter 1), we can decompose a Nest Thermostat into four high-level modules: main unit, base unit, local analytics features (Nest Sense), and cloud services. Therefore, R=4, and p_r, a_r, t_r represent the set of the physical components, analytics models, and time states for each module of the

[21] Tensor outer product uses the symbol "∘."

Nest Thermostat. We can also treat the Nest Thermostat family as a single tensor, $\mathcal{H}$. Then, the three generations can be seen as three decompositions. Hence, R=3, and p_r, a_r, t_r represent the set of the physical components, analytics models, and time states for each generation of the Nest Thermostat.

Definition 6: *sPH Tucker Decomposition.* The Tucker decomposition is a form of higher-order PCA (Principal Component Analysis). We define the *sPH Tucker decomposition* for $\mathcal{H} \in \mathbb{R}^{I\times J\times K}$ as a product of the orthogonal factor matrices $A \in \mathbb{R}^{I\times P}$, $B \in \mathbb{R}^{J\times Q}$, $T \in C^{K\times R}$ multiplied by a core tensor $\mathcal{G} \in \mathbb{R}^{P\times Q\times R}$ which shows the level of interaction between the different components:

$$\mathcal{H} \approx \mathcal{G} \times_1 A \times_2 B \times_3 C = \sum_{p=1}^{P}\sum_{q=1}^{Q}\sum_{r=1}^{R} g_{pqr} a_p \circ b_q \circ c_r = [\![\mathcal{G}; A, B, C]\!] \tag{2.5}$$

Where $[\![\mathcal{G}; A, B, C]\!]$ is a shorthand representation introduced in Kolda (2006). Element wise, the Tucker decomposition is

$$h_{ijk} \approx \sum_{p=1}^{P}\sum_{q=1}^{Q}\sum_{r=1}^{R} g_{pqr} a_{ip} \circ b_{jq} \circ c_{kr} \tag{2.6}$$

Where $i = 1, \cdots, I, j = 1, \cdots, J, k = 1, \cdots, K$. If *P*, *Q*, *R* are smaller than *I*, *J*, *K*, the core tensor $\mathcal{G}$ can be thought of as a compressed version of $\mathcal{H}$.

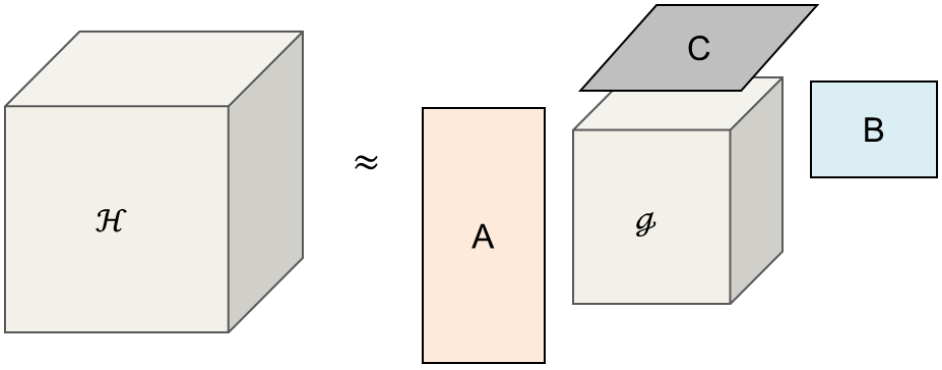


Figure 2.19 sPH Tucker decomposition

It is noted that Tucker decompositions are not unique. This freedom allows practitioners choosing transformations that simplify the core structure in some way so that most of the elements of $\mathcal{G}$ are zero, thereby eliminating interactions between corresponding components and improving uniqueness. Therefore, Tucker decomposition can be used to describe a scale-based product architecture, within

which each key component can implement integral architecture to minimize the number of components and interfaces. Graphically, it is shown in Figure 2.19.

Example. The Nest Thermostat's Auto-Schedule feature can be decomposed as an integral architecture including necessary physical components, analytics models, and discrete time states with three orthogonal factor matrices to describing the configuration of user environment, user behaviors, and the thermostat installation/running states. This way, the Auto-Schedule core module can be configured to accommodate individual use scenarios.

2.3 Research Method: *A Smart Products Lifecycle Management (sPLM) Framework*

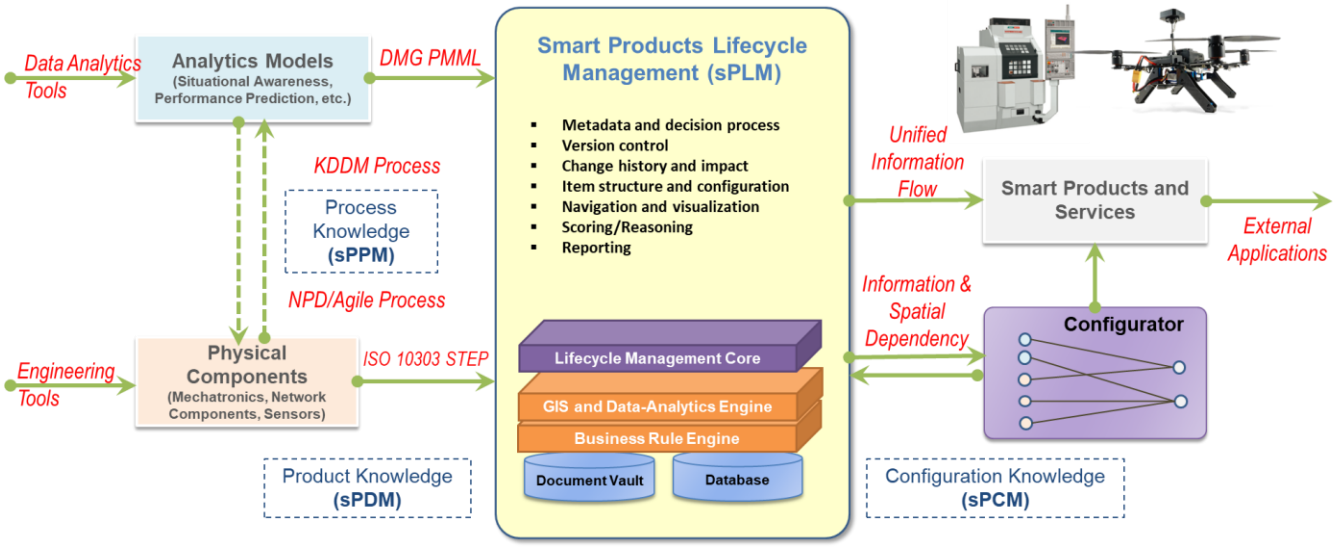


Figure 2.20 A Smart Product Lifecycle Management (sPLM) system architecture

Based on the study of literatures and the smart products hypercube decomposition, the overall method to guide the remaining discussions is depicted in Figure 2.20. A Smart Products Lifecycle Management (sPLM) framework is being implemented for validating the concepts and methodologies proposed to address the problems and questions discussed in Chapter 1.

As shown in Figure 2.20, the sPLM shall include standard lifecycle management functions (versioning, change management, user roles, authentication) commonly for physical products, analytics

models, and smart products. It also should have connectors (or translators) for authoring tools used by engineers (e.g. CAD software) and by data scientists (e.g. data analytics tools). This will enable them to access and exchange multidisciplinary data and models so as to construct smart products data. The sPLM shall also embed rule-based engine and scoring engine for knowledge elicitation, reasoning, knowledge model interpretation, and model real-time execution.

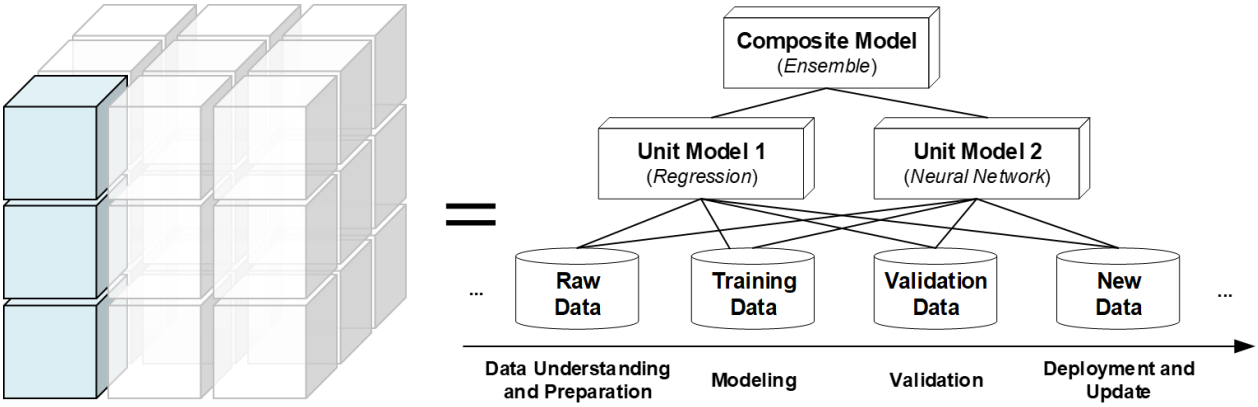
Three types of knowledge, product-, process-, and configuration knowledge, are needed to be elicited, modeled, and mapped at different abstract levels. In the sPLM framework, we will develop:

- *Product Data Model for Smart Products* (termed as "sPDM**"**): sPDM will uniformly represent and compose physical products models created by engineers and analytics models created by data scientists. This needs to be done by extending and bridging the semantic models (e.g. ISO STEP and PMML) already developed in the engineering and data science communities.
- *Process Model for Smart Products Development* (termed as "sPPM"): sPPM will facilitate modeling the activity interaction points, information flows, and information content during the co-development of physical components and analytics models, establishing a close understanding of interfaces between engineers and data scientists in an integrated product development team.
- *Configuration Model for Smart Products* (termed as "sPCM**"**): sPCM will facilitate the product definition and modular design of physical product architecture and analytics model architecture, so that modular physical components and analytics models can be uniformly composed on demand to accomodate the faster evolution of analytics models.

The three models – sPDM, sPPM, and sPCM – form the foundation of a Smart Products Lifecycle Management (sPLM) platform. In the following chapters, the methodologies and issues related to develop the three models are elaborated and illustrated using examples.

Chapter 3

A-T Space: *Modular Design of Data Analytics and Its Lifecycle Issues*



Let's first look at the A-T space of the Smart Products Hypercube. How can an analytics model be treated as a product? If this can be done, it would transform conventional task-oriented data analytics activities into a data products development process. Then, can we apply or adapt the methodologies existing in traditional manufacturing domain to the data products? In this chapter, we systematically investigate the issues related to the product definition, standardization, modular design approaches, and lifecycle management requirements for analytics models.

3.1 Product Definition of Analytics Models

3.1.1 What Is Analytics?

According to the Merriam-Webster dictionary, *Analysis*[22] is "a detailed examination of anything complex in order to understand its nature or to determine its essential features: a thorough study;" while

[22] Merriam-Webster: Analysis, https://www.merriam-webster.com/dictionary/analysis

Analytics[23] is defined as "the method of logical analysis," which is the discovery, interpretation, and communication of meaningful patterns in data. Analytical methods use data to answer questions that occurred in the past (*what happened* and *why did it happen*), but also provide insights or deductive reasoning to act in the future (*what will happen next* and *what should we do*). Nowadays, a hybrid modeling technique involving analytical and numerical methods, coupled with empirical data and artificial intelligence techniques has been applied to scientifically quantify the influence of parameters in a physical system (Jawahir et al., 2007). As shown in Figure 3.1, model-based techniques have been utilized to generalize computation, prediction, and optimization involved in different levels of analytics including descriptive analytics, predictive analytics, and prescriptive analytics.

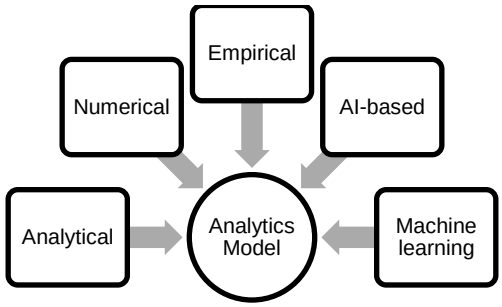


Figure 3.1 Classification of analytics model (adapted from Kim et al., 2015)

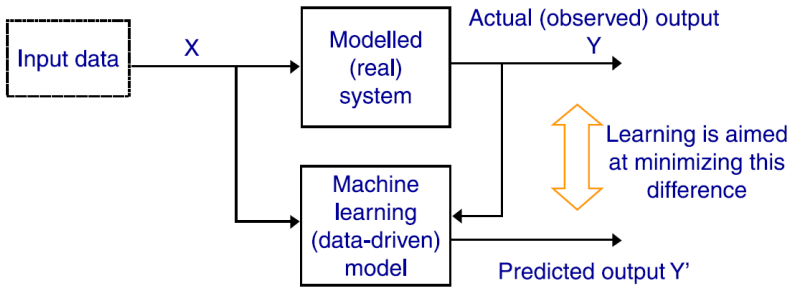


Figure 3.2 Complement knowledge-driven models with data-driven modeling (Solomatine and Ostfeld, 2008)

More specifically, in traditional analytical and numerical modeling and simulation, the functional

[23] Merriam-Webster: Analytics, https://www.merriam-webster.com/dictionary/analytics

relationships used in the equations consist of the first-principle laws (e.g. law of conservation of mass, Darcy's law, thermodynamics and energy conservation). These functional relationships are deterministic and unchangeable. However, there are some physical phenomena that are too complex to be modeled. We may not know all the parameters that are involved in the behavior of a phenomenon. Even if we know all the parameters, the relationships between these parameters may be too complex to model.

Data-driven modeling is based on analyzing the data about a system, to find connections between the system state variables (input, internal, and output variables) without explicit knowledge of the physical behavior of the system (Figure 3.2), which greatly expand the conventional empirical modeling and numerical modeling. It is beneficial in situations where there is a considerable amount of data available but it is difficult to build adequate knowledge-driven models (Solomatine and Ostfeld, 2008).

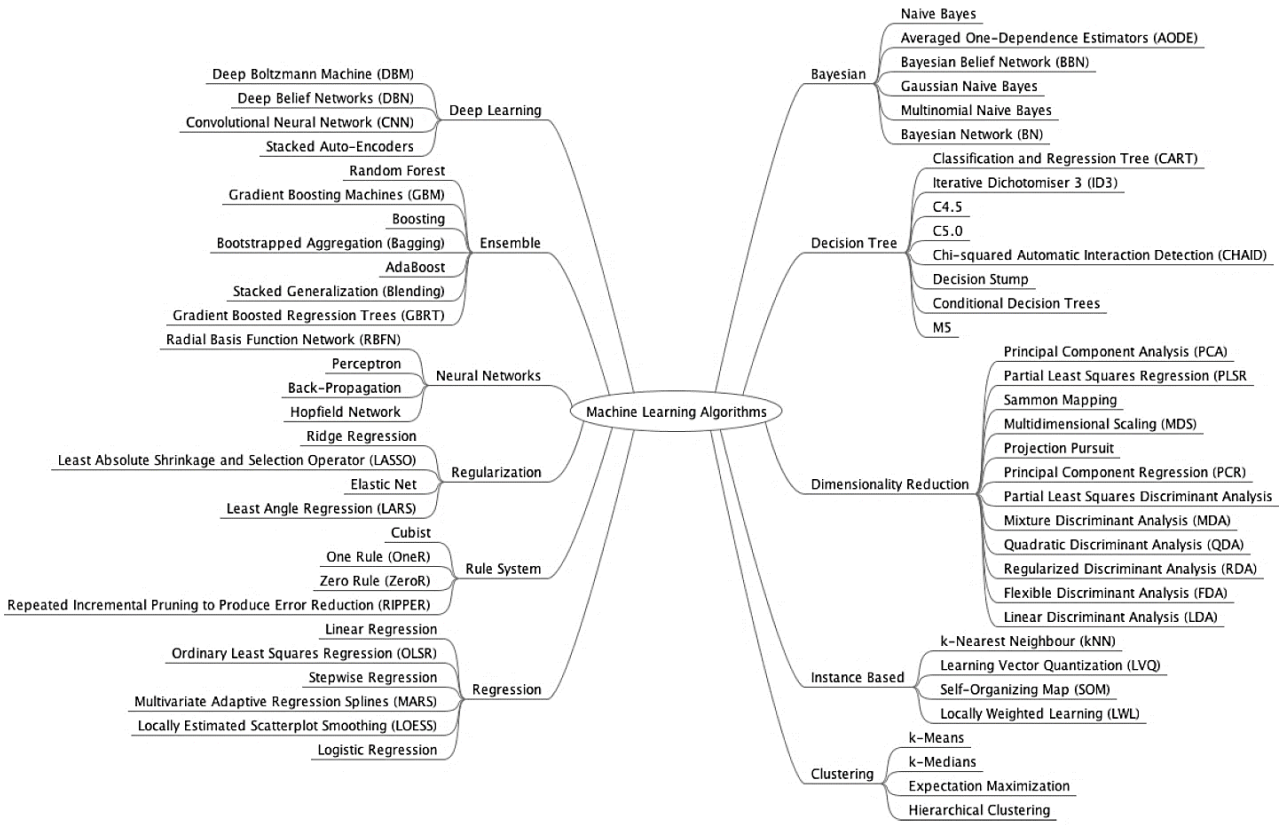


Figure 3.3 Machine learning algorithms and classifications (Chatterjee, 2016)

Numerous machine learning (ML) algorithms are available for creating data analytics models (Figure 3.3). They can be classified in three different types, namely *Supervised Learning*, *Reinforcement Learning*, and *Unsupervised Learning*. They differ in how feedback is provided: supervised learning uses labeled data (i.e. correct answer is given); unsupervised learning uses unlabeled data (i.e. no answer is given); while in reinforcement learning, feedback includes how good the output was but not what the best output would have been. Recently, *Deep Learning* has become a popular and successful approach. Neural Networks with many layers of nodes and large amounts of data are the basis of deep learning. Each added layer represents knowledge or concepts at a level of abstraction that is higher than that of the previous one.

3.1.2 Analytics Model Architecture

For real-world problems, different analytics models are needed to be ensembled or composed to achieve the necessary performance. For instance, combining classifiers can (1) avoid the worst classifier by averaging several classifiers (from *Statistical* perspective); (2) improve the performance of the best individual classifier (from *Representational* perspective); and (3) stabilize the optimization process (from *Computational* perspective) (Dietterich, 2000). And, a unified deep learning model to solve tasks across multiple domains (image captioning, speech recognition, language translation, etc.) may facilitate transfer learning (Kaiser et al., 2017).

As shown in Figure 3.4, ensemble methods can be categorized as *generative* and *non-generative* strategies. The non-generative strategy includes *ensemble fusion* and *ensemble selection* methods. Both use a predetermined set of learning machines previously trained with suitable algorithms. The base learners are then put together by a combiner module that may vary depending on the requirement of the output of the individual learning machines. The ensemble fusion (integration) assumes that all classifiers contribute to the final decision; the ensemble selection choses one classifier to give the final decision to

each pattern, assuming that classifiers are complementary.

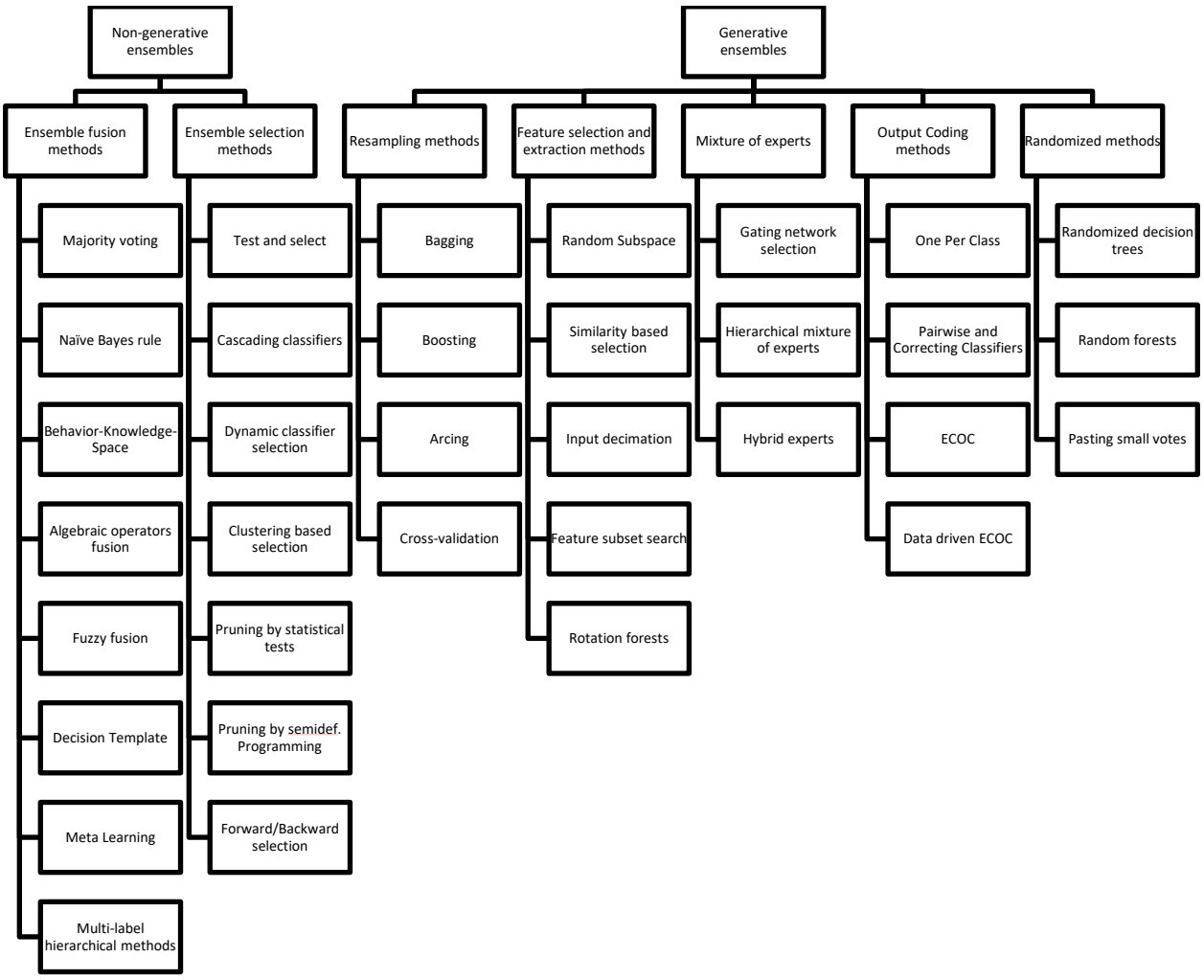


Figure 3.4 Classification of ensemble methods (Re and Valentini, 2012)

On the other side, the generative ensemble methods are able to generate base learners by acting on the base learning algorithm or on the structure of the dataset in order to actively boost *diversity* and *accuracy* of the base learners. Here, the classification accuracy is defined as the percentage of correct predictions made by the learner algorithm on the test data; and diversity is the maximum difference that can be produced in the selection of the subset of the training set of data for training and building each classifier. These ensemble methods can perturb the structure and the characteristics of the available input data, as in *resampling* methods or in *feature selection/subsampling* methods, or can manipulate the

aggregation and the coding of the classes (*Output Coding* methods), or can select base learners specialized for a specific input region (*mixture of experts* methods). They can also randomly modify the base learning algorithm, or apply randomized procedures to the learning processes to improve the diversity or to avoid local minima of the error (*randomized* methods).

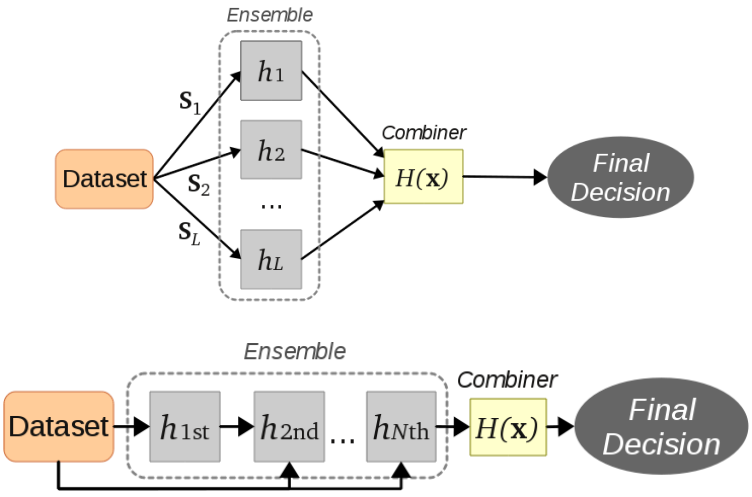


Figure 3.5 (a) Parallel architecture; (b) Serial architecture (Ponti, 2011)

Model structure wise, an ensemble model typically consists of some models, a fusion function to fuse all the models, an aggregation method to aggregate the results, and the overall framework architecture to connect those elements. Let h_i be the i^{th} classifier on an ensemble of *N* classifiers, $S = (x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)$ be the training data set, S_i be a version of the available dataset, and *H(x)* be a function that combines/selects the decisions of various learners about a new input pattern *x*. It is possible to design an ensemble system into two main architectures (Ponti, 2011):

- *Parallel architecture*: with a representation showed in Figure 3.5a, the set of classifiers are trained in parallel, and their output are combined afterwards to give the final decision. Parallel architecture is more generic for many applications.
- *Serial architecture*: with a representation in Figure 3.5b, a primary classifier is used, and when it is not able to classify some new pattern by rejecting it, we use a second classifier that is trained in order to be accurate on the errors of the previous classifier. A third and fourth classifier can be

used and so on. Sequential methods are often application-specific, and are also useful in the context of on-line learning.

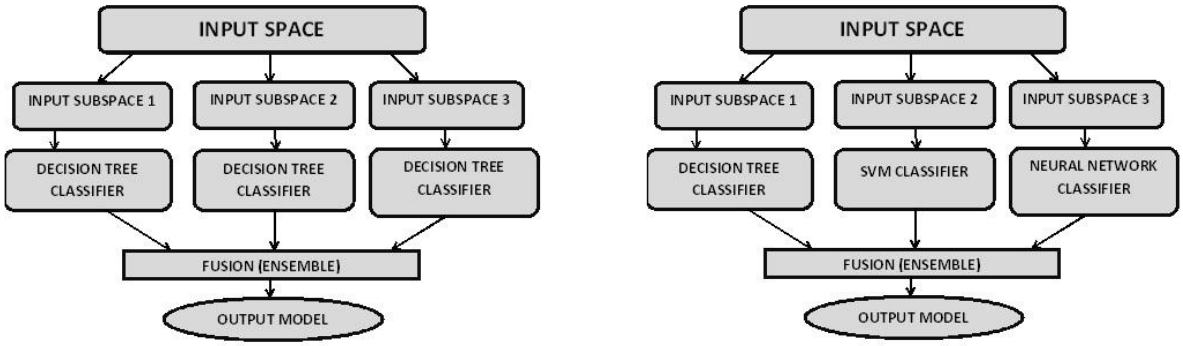


Figure 3.6 (a) Different classifiers in ensemble; (b) Same classifier in ensemble (Asmita and Shukla, 2014)

In the parallel architecture, these classifiers can be the same for all the input subsets (Figure 3.6a shows the classifiers are all decision tree models) or may be different for each input subset (Figure 3.6b shows that the classifiers can be decision tree, support vector machine, or neural network model).

3.1.3 Requirements and Structure Representations for Analytics Models

Considering data as the raw "material", data-driven modeling can be seen as a production process for producing data products, delivering data or delivering results based on data. An analytics model is the product of this process. An analytics model is also an information-processing unit to produce decisions, taking data as inputs and generating certain level of decisions. Any large computation should be split into a collection of small, nearly independent, specialized sub-processes (Coltheart, 1999). Making decisions modular and explicit ensures an effective separation of concerns and a more streamlined design (Taylor et al., 2013). Simpler processes can be easily changed and updated. This increases the capacity for change built into a process and allows for a stable process even when decision-making is constantly changing and evolving.

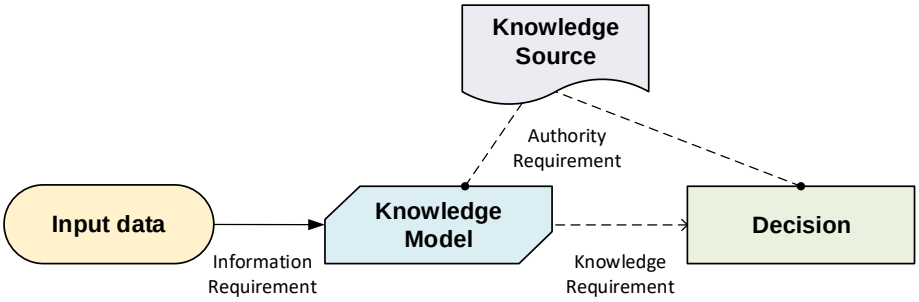


Figure 3.7 DMN for decision requirement representation (Li et al., 2017b)

The Decision Model and Notation (DMN)[24], developed by the Object Management Group (OMG), provides a graphical representation for decision requirements. The DMN notation creates a standardized bridge for the gap between the business decision design and decision implementation. As shown in Figure 3.7, a decision requirement can be represented as four elements: (1) *Input Data* corresponding to a concept of data; (2) *Decision* element corresponding to the decision that determines the output value from a number of input values through some decision logic; (3) *Knowledge Model* corresponding to the function that encapsulates an area of domain knowledge as executable decision logic, possibly expressed as business rules, an analytics model, or an algorithm; and (4) *Knowledge Source* defining an authority for decisions or knowledge models. The different types of arrows indicate the information requirement, knowledge requirement, and authority requirement among the different types of elements, respectively.

The structure or the core decision logic of an analytics model can use expressive languages for specifying details that are readable and executable to humans and computers. The data analytics community has developed open standards to facilitate model exchange among different tools and automate model execution in data scoring engines. This provides the possibility to separate model production and model consumption (Li et al., 2015a). One well-adopted standard is the Predictive Model

[24] Object Management Group: DMN, https://www.omg.org/spec/DMN/

Markup Language (PMML)[25] developed by the Data Mining Group (DMG). A predictive analytics model conforming to the PMML specification contains key elements including (1) a *Header* element that contains general information about the predictive model; (2) a *DataDictionary* element that contains schema definitions for all the possible data fields used by the model; (3) *TransformationDictionary* and/or *LocalTransformations* elements that allow users to map data into a more desirable form to be used by the mining model; and (4) One or more model (*Model* or *MiningModel*) elements that define the structure and scoring method of a mining model.

The *DataDictionary* contains definitions for fields as used in mining models. It specifies the data types and value ranges. These definitions are assumed to be independent of specific data sets as used for training or scoring a specific model. A DataDictionary element can be shared by multiple models through the MiningSchema element related to an individual model. A DataDictionary element is also used in statistics and other information related to the training dataset that produces the model.

Table 3.1 The models available in PMML

Category	*Model Name*	*PMML Model Name*
Association	Association Rules	AssociateionModel
	Sequence Rules	SequenceModel
Clustering	Clustering	ClusteringModel
	Nearest Neighbors	NearestNeighborModel
Regression	Regression	RegressionModel
	General Regression	GeneralRegressionModel
Classification	Neural Network	NeuralNetwork
	Naïve Bayes	NaiveBayesModel
	Decision Tree	TreeModel
	If-Then Rule Set	RuleSetModel
	SVM	SupportVectorMachineModel
Others	Scorecard	Scorecard
	Change Detection	BaselineModel
	Time Series	TimeSeriesModel
	Text Mining	TextModel
	Model Ensembles	MiningModel

[25] Data Mining Group: PMML, http://dmg.org/pmml/v4-3/GeneralStructure.html

It is noted that many models created by the ever-increasing new algorithms have not or may not be standardized. However, similar to the physical-product world, not every model is necessary to be standardized because it may only be used in a limited way. The current PMML standard has covered a broad type of models that can be individually used or be assembled to synthesize more complex models (Table 3.1).

Each analytics model consists of a mining schema based on the type of model it represents, the detailed model structure, target fields and values, and output elements such as measures of accuracy. For instance, as shown in Figure 3.8, a *Ruleset* model consists of a number of rules, and each rule contains a predicate statement and a predicted result. A *Scorecard* model can map input attributes to a series of reason codes which provide explanations of each individual score. A *BayesianNetworkModel* has a network linked by two types of nodes: *discrete* and *continuous*. A discrete node has a limited set of possible values and a continuous node has a continuous value represented by a continuous distribution.

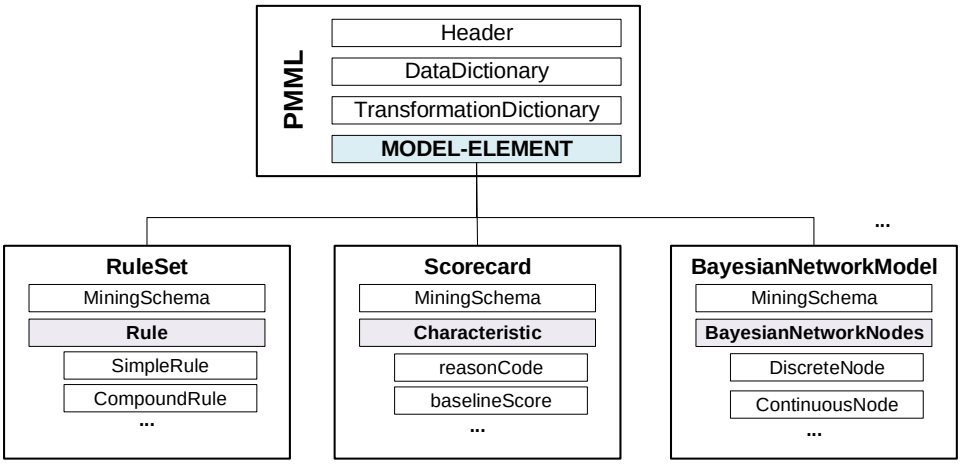


Figure 3.8 PMML for predictive model representation (Li et al., 2017b)

The *MiningSchema* element is the gate keeper of each model element. It allows each sub model in a model can uses a subset of the features. The *Segmentation* element allows users to represent different models for different data segments and also implement model ensembles and model sequences. These two elements control the input data subsets and feature subsets. The model combination methods

include: *majorityVote*, *weightedMajorityVote*, *average*, *weightedAverage*, *median*, *max*, *sum*, *selectFirst*, *selectAll*, and *modelChain*. These methods provide aggregation, composition, and manipulation operators for the sub models in an analytics model that can be designed as parallel, sequential, or hybrid architecture.

The two levels – requirement and structure/logic – together provide explicit and precise specification to allow automatic validation and/or execution of an analytics model.

3.1.4 Lifecycle of Analytics Models

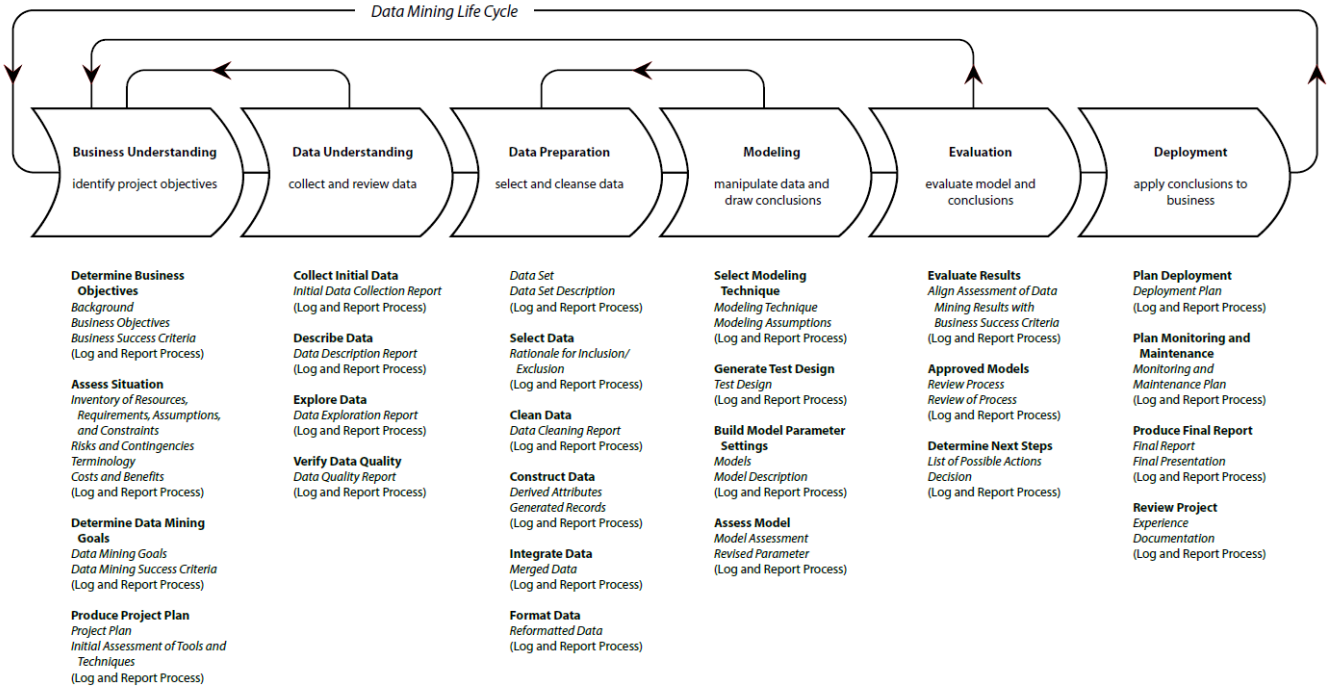


Figure 3.9 The CRISP-DM lifecycle model for KDDM (adapted from Shearer, 2000)

Formal process models for data analytics projects originated from the knowledge discovery and data mining (KDDM) community. CRISP-DM (CRoss-Industry Standard Process for Data Mining) is one of the more successful process models that have been adopted by both industry and academia (Kurgan and Musilek, 2006). CRISP-DM is a waterfall model that prescribes six high-level phases to formally describe a data analytics project and each phase is further decomposed into several key tasks

and deliverables (Figure 3.9). Efforts have been attempted to extend the CRISP-DM model for data mining engineering (Marbán et al., 2009) and machine learning engineering (Sapp, 2017).

The CRISP-DM's six high-level phases – *Business Understanding*, *Data Understanding*, *Data Preparation*, *Modeling*, *Evaluation*, and *Deployment* – well capture the necessary lifecycle stages for data science activities (Li et al., 2015b). More specifically, Business Understanding identifies the project objectives and requirements, and converting them into a data analytics problem definition and a preliminary plan. Data Understanding collects initial data, identifies data quality, and explores the data to form initial hypotheses. Data Preparation preprocesses the raw data to construct the final dataset. Modeling uses various modeling techniques to construct the model and optimize the parameters. Model Evaluation thoroughly reviews the steps to create the model, to ensure that the model achieves the business objectives. Model Deployment organizes and presents the knowledge gained to the stakeholders who use the model.

3.2 Modular Design of Analytics Models

We have introduced in Chapter 2 that two approaches, Module-based and Scale-based, are available for product family design. We explain below how these two approaches can be adapted for architecture design of analytics models.

3.2.1 Parametric (Scale-based) Design Approach for Analytics Models

Analytics models are highly data-dependent, frequently tuned and updated, making it difficult to be standardized and reused for different use environments. An analytics model might be discarded or archived once it accomplishes the desired task. For example, finance industry usually implements a *champion-challenger* strategy regarding the model development and deployment (chu et al., 2007). The champion is the production model (decision logic or business rules) that is based on data from several

previous time periods. Newer challenger models are built on data from more recent time periods. If the performance of a challenger model on a segment of data exceeds the champion model, it becomes the new champion (at least for that data segment). This process will continuously repeat in a dynamic business environment. In a high-throughput environment, thousands of models might be generated; the functional form is most likely fixed and the models are simply retrained.

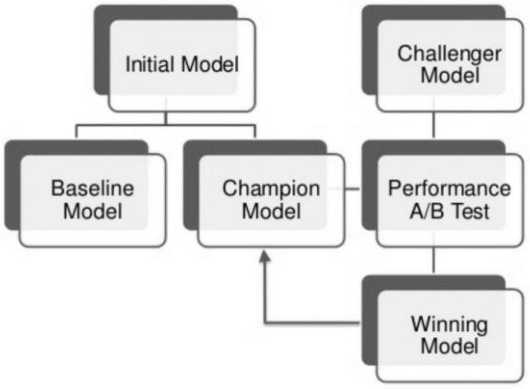


Figure 3.10 Champion/Challenger model used in business production environment (Liu, 2015)

The *Parametric* or *Scale-based* approach can be used for modular design of this type of model. In physical product design, the parametric approach mentioned in Chapter 2 allows one or more scaling variables to be used to stretch or shrink the product in one or more dimensions. An analytics model can be derived from a *parametric* form $P(x|\theta, D)$, where θ is a vector of a finite set of parameters, x is a vector of the future predictions, and D is the observed data. Figure 3.11 shows the decision requirement representation of this type of model.

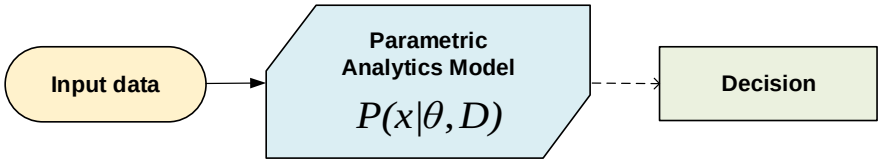


Figure 3.11 Parametric analytics model (adapted from Li et al., 2017a)

The base model structure keeps the same for all variations but the model parameters can be adjusted for optimal performances. For example, the PMML-encoded polynomial regression model shown as in

Table 3.2 is a parametric model for predicting the power energy consumption of a CNC machining center. Two instances can be derived from this model for producing parts based on two different types of materials: *aluminum* and *steel*. The two regression models have the same model structures with the same predictors but different predictor coefficients.

Table 3.2 A polynomial regression model expressed in PMML language (Li et al., 2017a)

```
<PMML version="4.1" xmlns="http://www.dmg.org/PMML-4_1">
  <Header copyright="sns8">
    <Application name="KNIME" version="2.9.2"/>
  </Header>
  <DataDictionary numberOfFields="5">
    <DataField name="FEEDRATE" optype="continuous" dataType="double"/>
    <DataField name="SPINDLE_SPEED" optype="continuous" dataType="double"/>
    <DataField name="CUTTING_DEPTH" optype="continuous" dataType="double"/>
    <DataField name="CUTTING_DIAMETER" optype="continuous" dataType="double"/>
    <DataField name="TOTAL_POWER" optype="continuous" dataType="double"/>
  </DataDictionary>
  <RegressionModel functionName="regression" algorithmName="PolynomialRegression" modelName="KNIME Polynomial
Regression" targetFieldName="TOTAL_POWER">
    <MiningSchema>
      <MiningField name="FEEDRATE" invalidValueTreatment="asIs"/>
      <MiningField name="SPINDLE_SPEED" invalidValueTreatment="asIs"/>
      <MiningField name="CUTTING_DEPTH" invalidValueTreatment="asIs"/>
      <MiningField name="CUTTING_DIAMETER" invalidValueTreatment="asIs"/>
      <MiningField name="TOTAL_POWER" invalidValueTreatment="asIs" usageType="predicted"/>
    </MiningSchema>
    <RegressionTable intercept="-0.11970484212343813">
      <NumericPredictor name="FEEDRATE" coefficient="-0.051039282222963445"/>
      <NumericPredictor name="FEEDRATE" exponent="2" coefficient="0.10165911010248263"/>
      <NumericPredictor name="FEEDRATE" exponent="3" coefficient="-0.04423234642719831"/>
      <NumericPredictor name="SPINDLE_SPEED" coefficient="0.49079258845095675"/>
      <NumericPredictor name="SPINDLE_SPEED" exponent="2" coefficient="-0.1752970995690415"/>
      <NumericPredictor name="SPINDLE_SPEED" exponent="3" coefficient="0.15959779389737339"/>
      <NumericPredictor name="CUTTING_DEPTH" coefficient="0.42510095004136295"/>
      <NumericPredictor name="CUTTING_DEPTH" exponent="2" coefficient="-0.34276761989748294"/>
      <NumericPredictor name="CUTTING_DEPTH" exponent="3" coefficient="0.1664911886467786"/>
      <NumericPredictor name="CUTTING_DIAMETER" coefficient="0.23046564162660843"/>
      <NumericPredictor name="CUTTING_DIAMETER" exponent="2" coefficient="0.11808414635737563"/>
      <NumericPredictor name="CUTTING_DIAMETER" exponent="3" coefficient="-0.10781611953177617"/>
    </RegressionTable>
  </RegressionModel>
</PMML>
```

3.2.2 Combinatorial (Module-based) Design Approach for Analytics Models

Analytics model can also be designed by combining other analytics models: unit models or component models. For example, a model to predict manufacturing operations for features of a prismatic part can comprise of two unit-models: a ruleset model and a decision tree model. The ruleset model is for non-hole features such as a *face*, a *slot*, or a *pocket*; the decision tree model is for hole features (Figure 3.12). These two unit-models can be composed using a global configuration rule (another ruleset

model). The *Combinatorial* or *Module-based* approach can be adapted for this type of models (Figure 3.13 shows its decision requirement). The Combinatorial approach allows product family members being instantiated by adding, substituting, and/or removing.

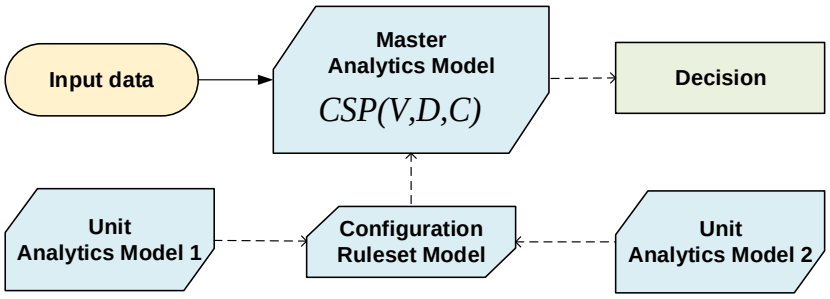


Figure 3.12 Combinatorial analytics model (adapted from Li et al., 2017a)

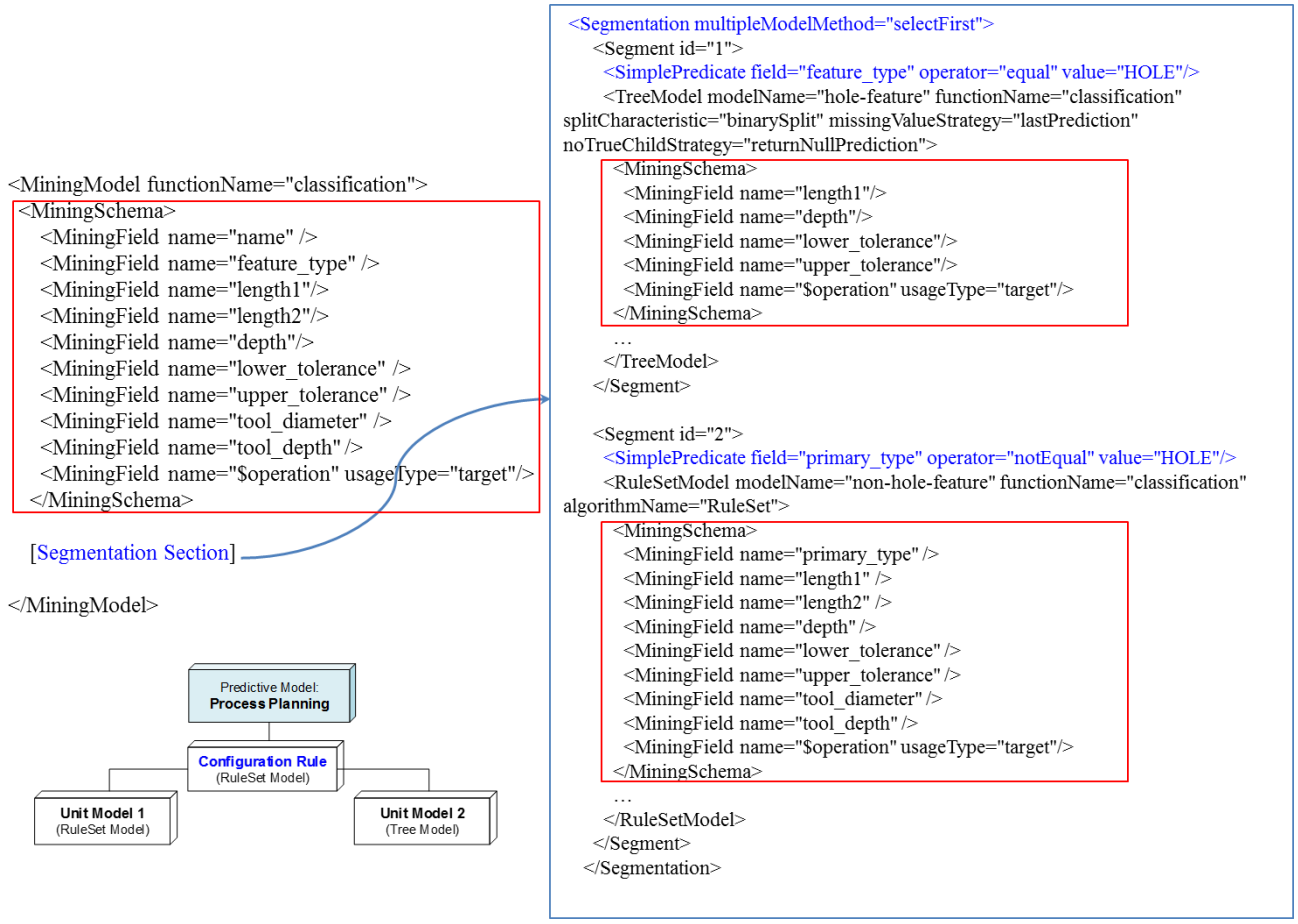


Figure 3.13 A manufacturing process planning model expressed in PMML language, consisting of a Ruleset model and a Tree model (adapted from Li et al., 2015a)

3.2.3 Modularity Quantification of Analytics Model Architecture

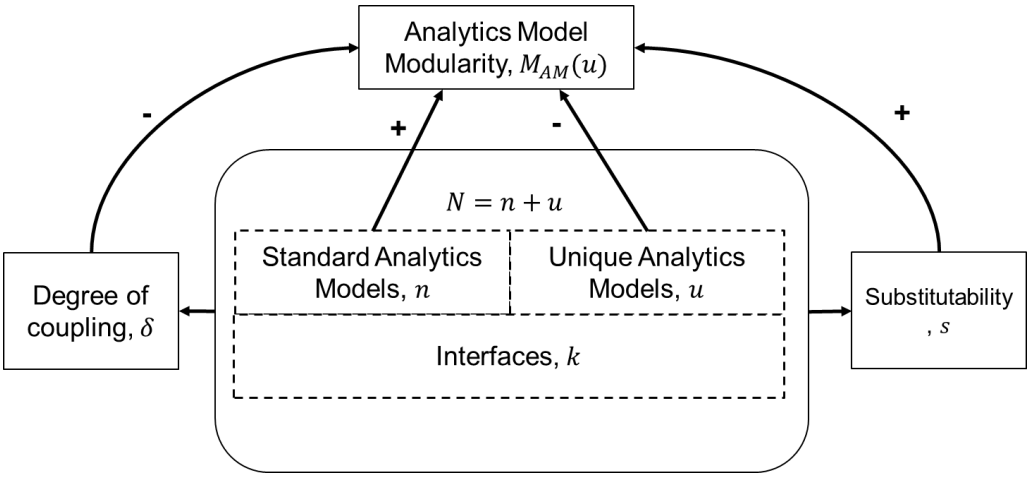


Figure 3.14 The factors contributing to modularity of an analytics model (The system dynamics diagram is following the approach by Mikkola, 2007)

As discussed in Section 2.1.3, Mikkola's modularity function, *M(u)*, integrates different factors affecting the modular performance of a product architecture (Mikkola and Gassmann, 2003). If an analytics model can be seen as a product, a similar modularity function for the architecture of the analytics model, $M_{AM}(u)$, can be derived. However, due to the interface between analytics models is information related to data and decisions, we have to define them properly.

The factors contributing to the modularity of an analytics model and their relationships are depicted in Figure 3.14. The assumptions for analytics model modularity are given as below, and the derivation details are shown in Table 3.3.

- An analytics model can be *standard (n)* or *unique (u)*.
- Standard analytics models are defined as the models that have been standardized in open or industrial standards. For example, the PMML has standardized regression model, decision tree model, neural network model, and ensemble of these standard models.
- Unique analytics models are defined as the ad-hoc models created for particular functions. These models are encoded as certain mathematical/statistical forms and cannot be exchanged without providing the original codes.

- An analytics model and its associated data construct the minimal analytics model component, because the analytics model is meaningless without the associated data definition. Thought of this way, a data itself can be seen as the simplest analytics model (*Y=X*). The data fields specification constructs the input interfaces of the analytics model.
- Decision is the consumer (user) of the analytics model. Single or multiple data fields specification for a decision constructs the output interfaces of the analytics model;
- Through compatible data fields, an analytics model can take the decisions from alternative analytics models as its inputs.
- The interfaces of an analytics model are defined as the incoming linkages from input data plus the outgoing linkages to decisions.
- The *degree of coupling (δ)* is defined as the ratio of the total number of incoming links by the number of analytics model in a given subsystem:

$$\delta = \frac{the\ total\ number\ of\ incoming\ links}{the\ number\ of\ analytics\ models} \tag{3.1}$$

- The *substitutability (s)* is defined as the number of decisions an analytics model can make (the outgoing links) by the average number of interfaces of unique analytics model components.

$$s = \frac{the\ number\ of\ outgoing\ links}{the\ average\ incoming\ links\ of\ unique\ models} \tag{3.2}$$

A modified modularity function for the analytics model architecture, $M_{AM}(u)$, is then defined as a function of variables: the number of unique models (u), the total number of models (N), the substitutability (s), and the degree of coupling (δ):

$$M_{AM}(u) = e^{-\frac{u^2}{2Ns\delta}} \tag{3.3}$$

The mathematical form of this function is similar as Mikkola's function for physical products but the variables have different meanings.

Table 3.3 The derivation of $M_{AM}(u)$:

The total number of analytics models, *N*, and the proportion of unique analytics models, *b*, present in a given data product architecture are

$$N = n + u;\ \ b = \frac{u}{N}$$

Let's assume there is a relationship between the degree of modularization, *M*, and the number of unique analytics models, u, $M_{AM} = f(u)$. The lower the number of unique analytics models, the higher the degree of modularization. A perfect modular data product architecture has no unique analytics models, i.e. $b = 0$.

Let's define the degree of modularization, M_{AM}, decreases at a rate, r, proportional to the amount of modularization present in each set of unique analytics models, *u*. Then, the amount of modularization changes by the amount of $\Delta M_{AM} = rM_{AM}$ as the number of *u* varies. For any unit change of unique analytics models, the corresponding amount of modularization change ΔM_{AM} is proportional to the initial level of modularization, i.e.

$$\Delta M_{AM} = (-rM_{AM})\Delta u; r = \frac{b}{s\delta} = \frac{u/_N}{s\delta}$$

Here, the degree of coupling, δ, measures the tightness of coupling of a given data product architecture. The higher the value, the lower the degree of modularization embedded in the product architecture. The substitutability, *s*, measures the number of decision types (it stands for the product families that can consume the decisions) made involving the unique analytics models. The higher the value of *s*, the greater is the degree of modularization. $s\delta$ can be interpreted as the cumulative interface constraint effect of subsystems (or sub models) across product families. Therefore,

$$\Delta M_{AM} = (-rM_{AM})\Delta u = \left(-\frac{u/_N}{s\delta}\right)M_{AM}\Delta u, or$$

$$\frac{dM_{AM}}{du} = -\frac{u}{Ns\delta}M_{AM}$$

Solving this differential equation for any constant r, we get

$$M_{AM}(u) = M_{AM,0}e^{-\frac{u^2}{2Ns\delta}}$$

For u=0, which means the product architecture is perfectly modular, $M_{AM}(0) = M_{AM,0} = 1$, this is the initial condition. Thus,

$$M_{AM}(u) = e^{-\frac{u^2}{2Ns\delta}}$$

Now, let's illustrate how to apply this function to evaluate the architecture of an analytics model. as

shown in Figure 3.15, the Nest Thermostat's auto-schedule feature can be seen as a modular architecture that includes Time-To-Temperature, Auto-Away, Sunblock sub modules. Each sub module can independently make its own decision based on the fusion of sensor data and external service data. For example, the Sunblock module uses the built-in light sensor to track the sun's patterns and the temperature sensors to detect the heat spikes that occur in direct sunlight. It also considers sunrise and sunset schedule for the user's location. For an installed thermostat, the location is fixed, so the machine learning model for this location can be seen as a relatively static model, i.e. it can be a standardized model that can be shared by all the thermostat instances installed in that region. Its modularity is 1. However, the Time-To-Temperature and the Auto-Away modules have to take the user behavior into consideration and they need to adapt to each individual user's living pattern. Thus, we treat these two modules as unique analytics models.

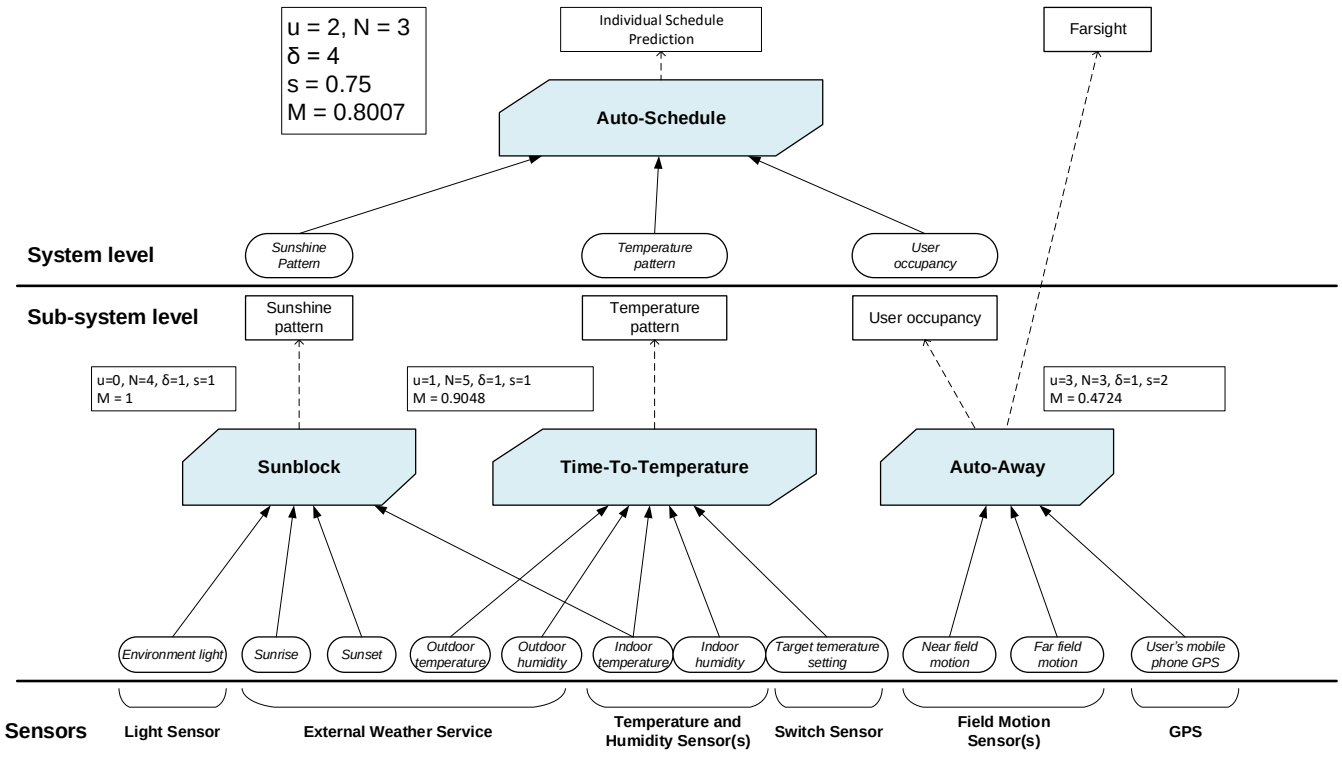


Figure 3.15 Modularity evaluation of Nest Thermostat's Auto-Schedule

Thought of this way, the Auto-Schedule feature now has two unique models out of three analytics

models, thus *u*=2 and *N*=3. The overall structure has *4+5+3=12* incoming links in total, the degree of coupling $\delta = 12/3 = 4$. The three sub modules predict *3* decisions (outgoing links) in total, the substitution factor $s = 3/(8/2) = 0.75$, where the two unique models have eight incoming links. Therefore, $M_{AM,Auto-Schedule} = e^{-u^2/2Ns\delta} = 0.8007$.

Similarly, we can calculate the modularity measures for the three sub modules. For the Time-to-Temperature model, only the *Target temperature setting* is counted as unique analytics model; for the Auto-Away module, all three inputs are user-specific so they are all treated as unique models. It is noted if we remove the three sub modules and let the Auto-Schedule module directly take all the sensor data as inputs, this results in $u = 4,\ N = 11,\ \delta = 11/11 = 1$, and $s = 1/(4/4) = 1$, thus, $M_{AM,Auto-Schedule} = e^{-u^2/2Ns\delta} = 0.4832$. That is, the modularity is reduced due to the increasing incoming interfaces. The modularity functions for both cases are plotted as shown in Figure 3.16.

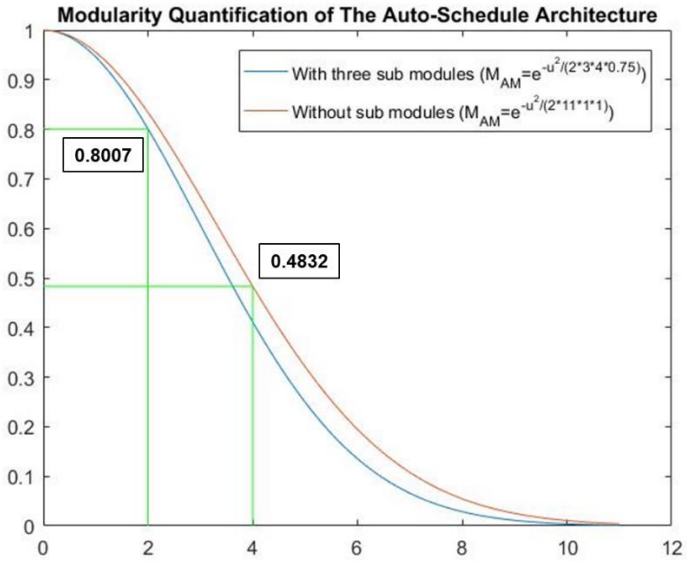


Figure 3.16 Modularity quantification of the Auto-Schedule architecture

3.3 Analytics Model Lifecycle Management

We have studied the issues related to the standardization of analytics models, the modular design

approaches, and the modularity quantification. Now, let's discuss how to implement these product design concepts from the analytics model's lifecycle perspective. In this section, we use an industrial product, a CNC (Computerized Numerical Control) machining center, to illustrate how to design and manage its analytics models. And, in next chapter, we will illustrate how to compose its physical components and analytics models. We use an open-source product lifecycle management (PLM) modeling platform, *Aras Innovator* (see Appendix B), to implement this case study. We demonstrate how to map the standard analytics model information model to the PLM data model to implement the analytics model lifecycle management functions.

3.3.1 Use Case Description: *A Smart CNC Machining Center*

$$\boldsymbol{TotalPower} = B_{00} + \sum_{k=1}^{3} B_{1k} \cdot \boldsymbol{FeedRate}^k + \sum_{k=1}^{3} B_{2k} \cdot \boldsymbol{SpindleSpeed}^k + \sum_{k=1}^{3} B_{3k} \cdot \boldsymbol{CuttingDepth}^k + \sum_{k=1}^{3} B_{4k} \cdot \boldsymbol{CuttingDiameter}^k \quad (3.4)$$

Table 3.4 Range of process parameters in machining (Li et al., 2015b)

Parameter	*Unit*	*Aluminum*	*Steel*
Feed rate	mm/rev	0.1 ~ 0.5	0.2 ~ 0.6
Spindle speed	rad/s	94 ~ 209	94 ~ 126
Cutting depth	mm	1 ~ 4	2 ~ 6

Energy consumption is an important performance indicator of industrial equipment. Power consumption is a measured scalar value, which is used to calculate the energy consumption by integrating power over machining time. Therefore, the ability to predict power consumption enables one to monitor the energy efficiency of a CNC machining center; and then control it proactively, if necessary. Let's consider a scenario in which the CNC machining center executes roughing operations to produce turned-parts from cylindrical work pieces. It produces a weekly batch of 350 parts made of aluminum and 150 parts made of steel. We collect time series data regarding three process parameters: *feed rate*, *spindle speed*, and *cutting depth* (Li et al., 2015b; Shin et al., 2016). Table 3.4 lists the ranges

of these process parameters. The dataset is used to train a cubic polynomial-based regression model as shown in the equation 3.4. This regression model has been encoded in PMML (see Table 3.2).

The regression model uses four machining process parameters as predictors: *feed rate*, *spindle speed*, *cutting depth*, and *cutting diameter*. Here, the cutting diameter is the outermost dimension of the work piece being turned, and can be calculated from the cutting depth. It is added to the model due to its influence on the cutting power (Shin et al., 2016). The process parameters vary depending on the work piece materials. Our two parts are made of two different materials: *Aluminum* and *Steel*. Thus, the resultant two unit-regression models have the exact same structures but different intercepts (B_{00}) and coefficients (B_{11}-B_{43}) for individual predictors of their regression equations. The two unit-regression models can be then individually used. They can also be composed into a single power prediction model, using material as a parameter to create the model configuration rule. Then, the power prediction model is used as a standard component by any compatible CNC machine product.

3.3.2 Modeling Analytics Models and Datasets

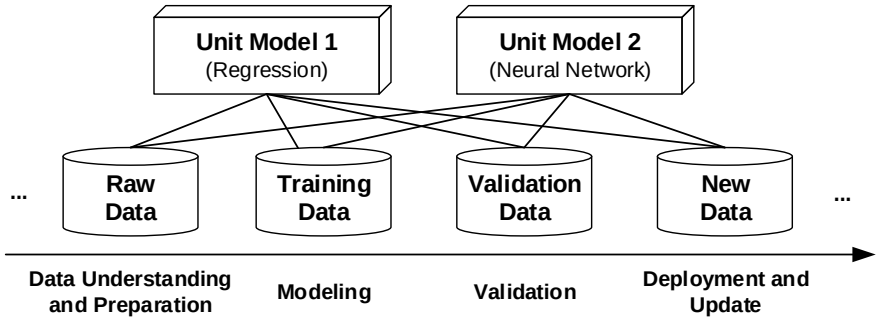


Figure 3.17 The relationship of unit predictive models and datasets over the model lifecycle (Li et al., 2015b)

In the view of analytics models, datasets are the primary "raw materials" used to create and validate an analytics model. A particular dataset can be used to create or validate many different unit analytics models. As an analytics model evolves over its lifecycle, so will its datasets. Therefore, the management functions must be able to track the changes in the datasets. To do this, composition relationships

between datasets and analytics models must be established (Figure 3.17). A more complicated analytics model can then be built by composing several unit models. Similarly, an analytics model is used to create a smart component, which is further used to build smart products (will be discussed in Chapter 4). This hierarchy of heterogeneous building blocks can be generalized using an *Item-Relationship-Item* structure, in which each item has its own unique identifier, properties and methods. Once such a modular structure is established, all building blocks can be accessed, used, traced and maintained consistently.

The hierarchy of the PMML elements shown previously (refer to Figure 3.8) can be mapped to the *Item-Relationship-Item* structure of the PLM system. For example, the root *PMML* element and the *DataField* element can be modeled as two item types, and can be connected with a *DataDictionary* relationship (see Figure 3.18 for a snapshot of the PMML schema modeling in PLM). Similarly, the *MiningSchema* is modeled as a relationship to connect the root *MODEL-ELEMENT* with the *MiningField* element. The *DataDictionary* and *DataField* schema defined in the current PMML specification can be used to formally model the dataset items. As noted, the composition of an analytics model and its relevant datasets is necessary to create and validate the model. The lifecycle stages and deliverables defined in the CRISP-DM reference model can be used to determine when an appropriate dataset should be attached to the analytics model.

Figure 3.19 shows the implementation of the regression model for aluminum material and its master PMML item to invoke the model. The PMML's elements are now organized conforming to the Item-Relationship-Item structure, i.e. each element's metadata has been collected and presented in a client form intuitively. This example PMML document has five data fields and one regression model. The regression model contains one regression table that consists of descriptive information of each predictor. Figure 3.20 demonstrates that the mining fields are reused in the two unit polynomial regression models with the same model structures but different coefficients for each predictor.

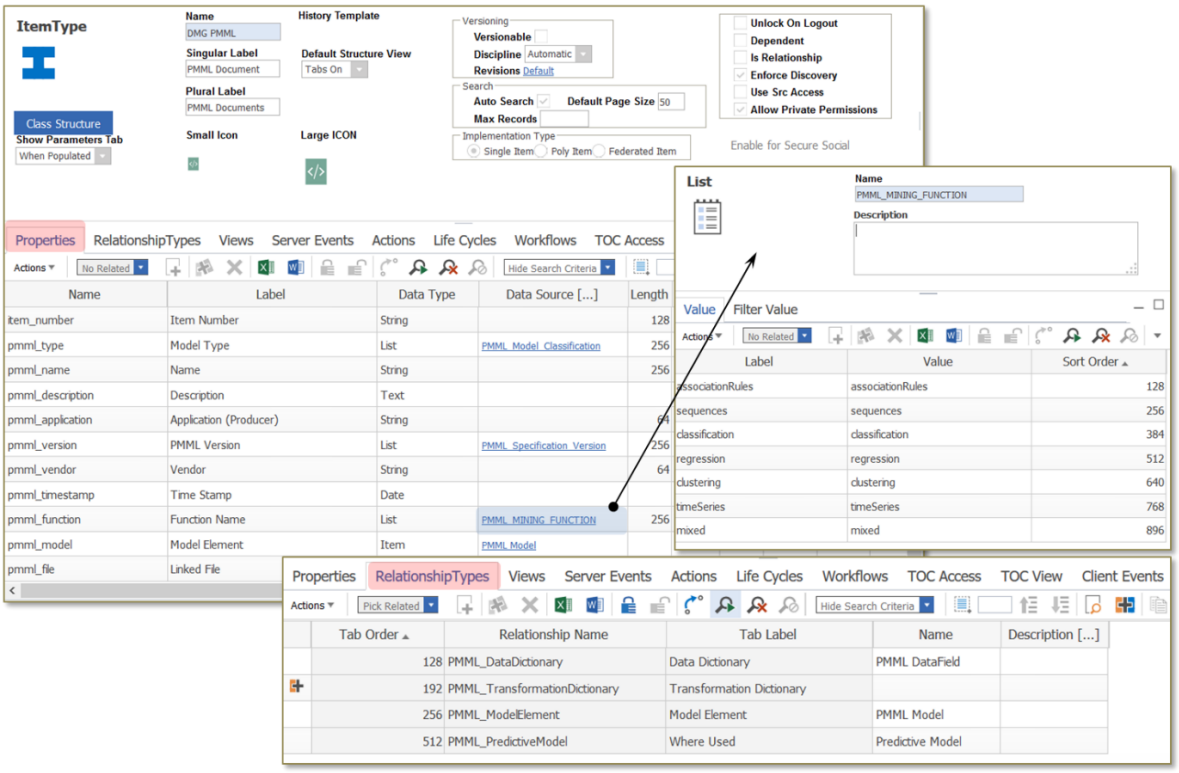


Figure 3.18 PMML Schema: Properties, Relationships, etc.

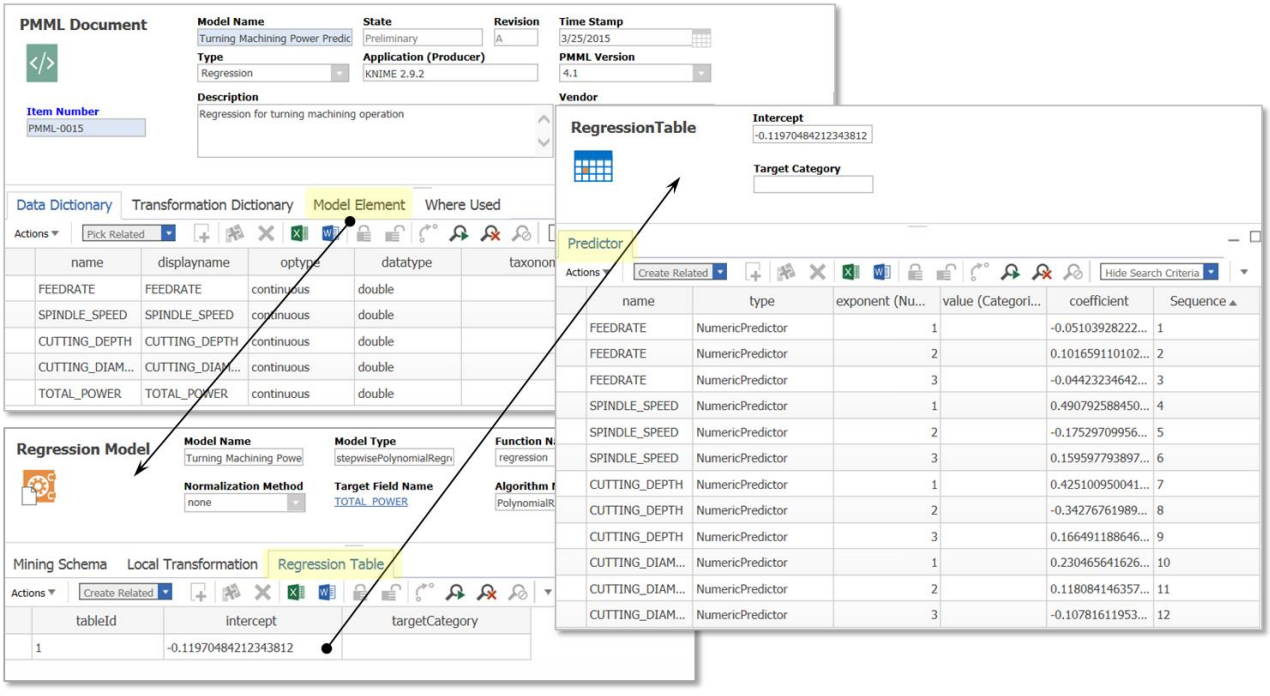


Figure 3.19 A PMML regression model's metadata modeled in the PLM system (Li et al., 2015b)

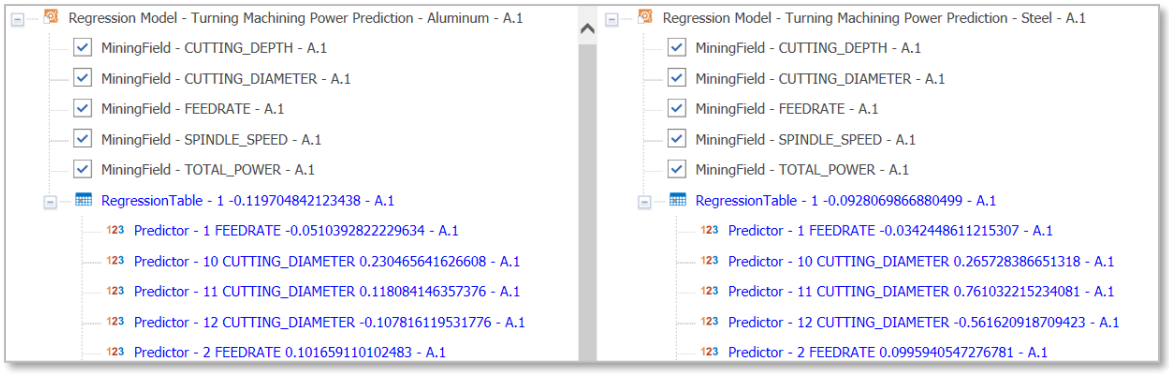


Figure 3.20 The comparison between the structures of two regression models (the left is for aluminum material and the right is for steel material) (Li et al., 2015b)

3.3.3 Modeling Composition of Unit Analytics Models

PMML supports model ensembles and model chains. The *Segmentation* element defined in PMML allows users to represent different models for different data segments. This element is also useful to encapsulate multiple models into a single PMML document. The elements defined in the PMML specification can be modeled as different items supported by their hierarchical relationships. For instance, the root *PMML* element has a relationship to a *MODEL-ELEMENT*. The *MODEL-ELEMENT* can be defined as a hybrid item in PLM so that it accepts different analytics models including regression models, neural network models, rule-set models, and so on. Each unit-analytics-model item can then be inserted into a master PMML item or multiple PMML items. This enables a given unit PMML model be reused as needed.

Configuration rules, which support the selection of a particular model to meet certain conditions, can be created as a separate ruleset model. This model can then be added into the master PMML item and also be linked to the involved unit models. In general, a master PMML item can contain all the necessary unit models plus at least one configuration ruleset model (where we assume all configuration rules can be decomposed and represented as IF-THEN rules to form a rule-set model). Figure 3.21 illustrates the conceptual structure of this composite predictive model.

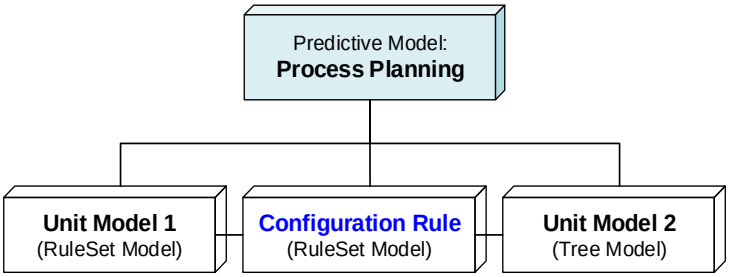


Figure 3.21 A predictive model including two heterogeneous unit models (Li et al., 2015b)

An analytics model can be modeled at different granularity levels. For instance, we can treat the whole predictive model as a single item with the metadata as its properties, and then attach a PMML document to the item. This is the traditional approach for document-centric data management. It is the simplest way to maintain the document information without losing any raw information because the original file can be referenced anytime as needed. The primary drawback of this approach is that those sub-elements within the PMML document cannot be accessed individually afterwards.

The other approach is to model the sub-elements of the PMML document as items. This approach increases the modularity and reusability of sub-elements that include data fields, data transformations, and models. As for example, two analytics models may use the same data fields but different algorithms; or, two polynomial regression models may have the same model structures with the same mining fields and different predictor coefficients. Modeling sub-elements provides a finer granularity of information for controlling the model structure. This allows users to take the full advantage of PLM functions, such as versioning, change management, and configuration management. However, a compromise must be made since there are additional costs for a finer data management: it means more data needs to be stored, and more complicated relationships need to be established and traced.

3.3.4 Modeling Lifecycle Stages and Activities

The phases and generic tasks defined in the CRISP-DM model (Figure 3.9) can be implemented in

the PLM system as one lifecycle map and several workflow maps. Detailed tasks and deliverables of each phase can also be modeled. Figure 3.22 shows a predictive model that has completed the *Modeling* phase, and is currently in the *Evaluation* phase of its lifecycle. Its version is labeled as *Alpha*, since it has been neither released nor deployed yet. The workflow model enables different participants to interact with one another following certain business rules. For illustration purpose, we assume the model development process involves two user-groups *DA* and *DM*, which stand for *Data Analytics* and *Data Mining* respectively, who have different roles in different lifecycle stages. A user in the DM group focuses on the activities and tasks during the modeling phase. S/he (1) requires the preprocessing work to be completed by other users in the DA group, and (2) submits the completed predictive model to appropriate users in the DA group for further post-processing.

The lifecycle model in the PLM system can also capture the appropriate relationships between the predictive model and its relevant datasets. For instance, the training dataset can only be used in the Modeling phase, and a newer training dataset may be used to update the predictive model during a *model-revision* process, which will trigger an ECN (Engineering Change Notice) process.

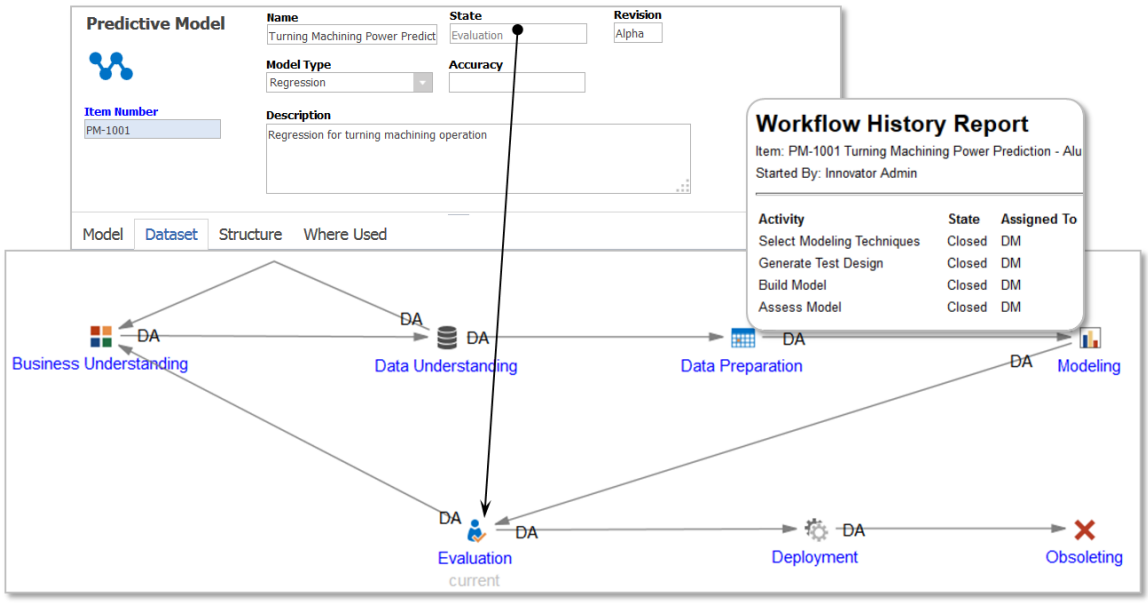


Figure 3.22 The lifecycle map of a predictive model (Li et al., 2015b)

3.3.5 Analytics Model Retrieval and Execution

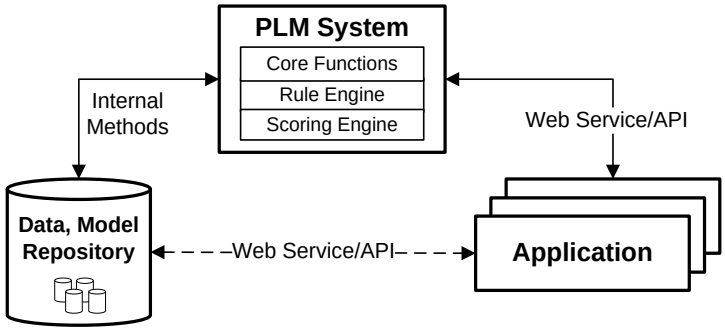


Figure 3.23 Analytics model retrieval and consumption (Li et al., 2015b)

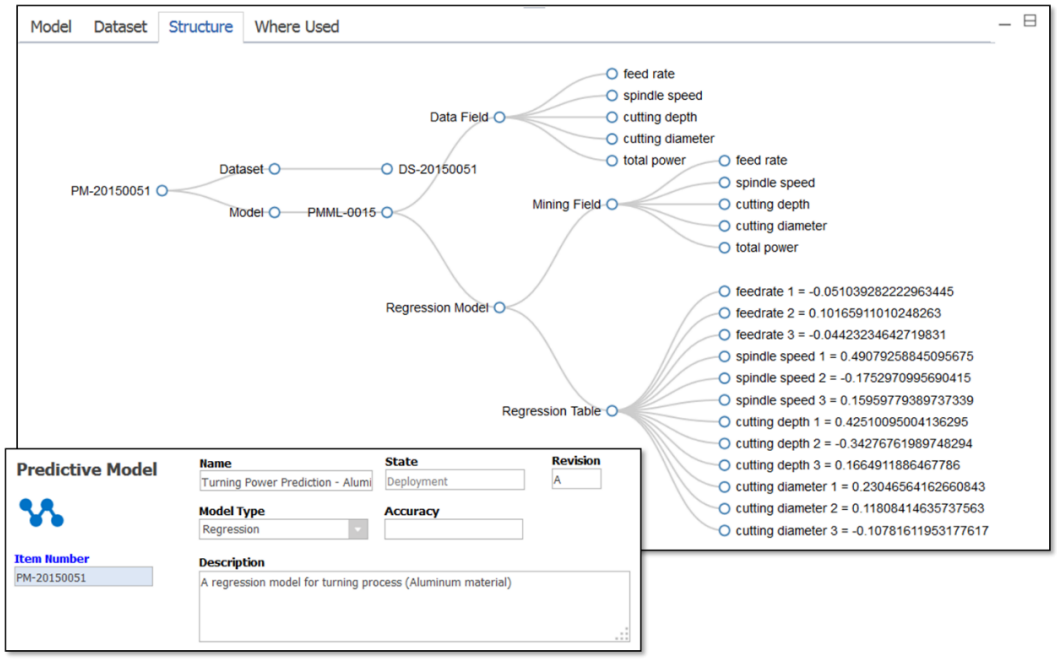


Figure 3.24 Visualization of the regression model structure

The PLM system now provides a repository for all available instance data and instance models. It also stores all the necessary lifecycle information such as states and revisions of the data and the models. An analytics model and its associated dataset can either be (1) executed inside the PLM system using PLM built-in execution engines (e.g. a PMML-based scoring engine, JPMML[26], for real-time prediction)

[26] JAVA PMML API, https://github.com/jpmml

and methods, or (2) retrieved and utilized by an external application through web services or application programming interfaces (API), as shown in Figure 3.23.

The data and models in the repository may also be exposed directly to external applications (see the dash line between the *Repository* block and the *Application* block, Figure 3.23). The PLM system should be capable of returning information regarding any dataset, unit predictive model, and any of their compositions, corresponding to different levels of queries requested. This modeling enables the traceability of the analytics models and their associative datasets (Figure 3.24) in case there is a need to trace the failure of a decision-making through an analytics model.

3.3.6 Data Analytics Tool Connector

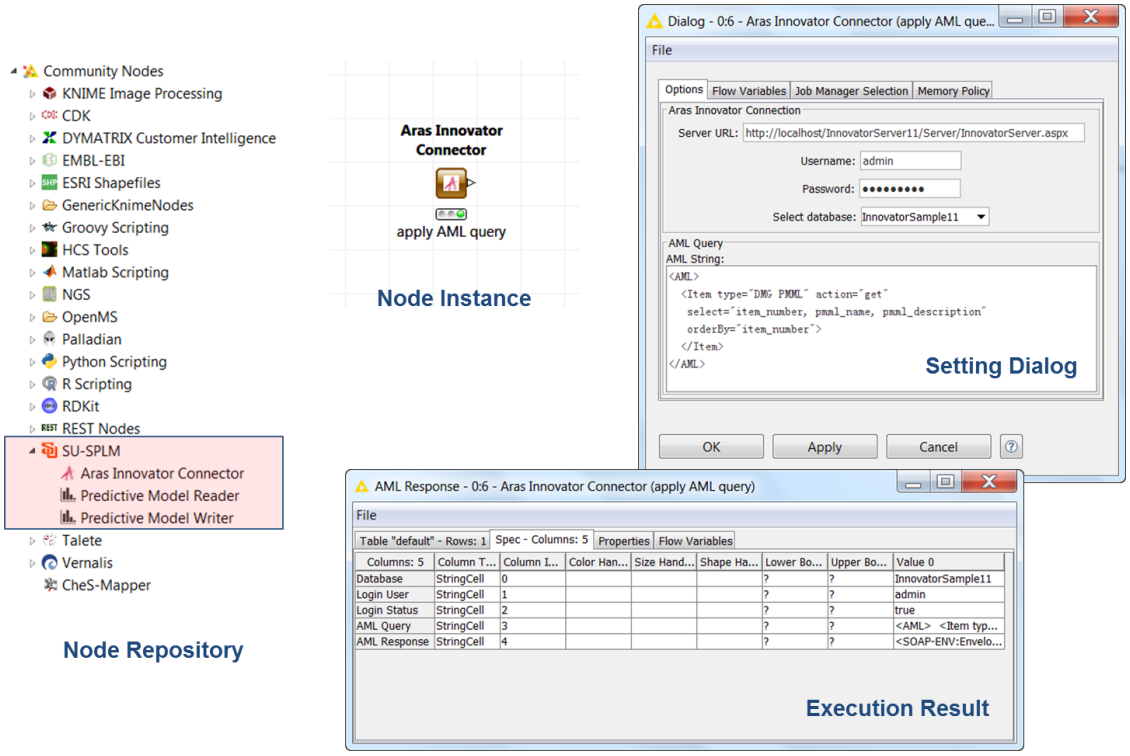


Figure 3.25 Connecting the PLM system and issuing query from KNIME analytics platform

PLM is an important collaboration platform to support all stakeholders including data scientists to access product-related information on demand. In order to enable data scientists to access the PLM

information from their familiar data analytics tools, A web-service-based PLM connector node has been developed for the KNIME analytics platform[27]. KNIME users can retrieve/add/update any authorized information in PLM by issuing appropriate queries, using the Aras Markup Language (AML, see Appendix B) provided by Aras Innovator. For instance, the user can query the predictive models developed by other users and then reuse the interested models.

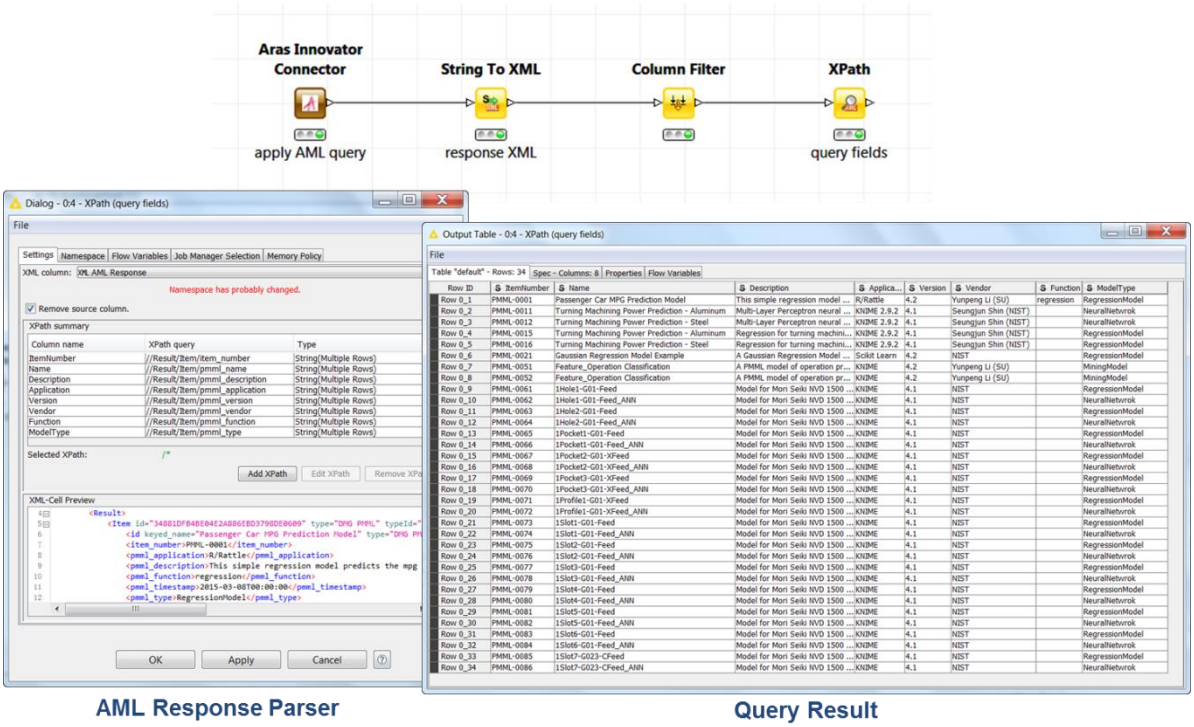


Figure 3.26 The response from Aras Innovator PLM

3.4 Summary

The advantage of modular design versus integral design of physical product architecture has been well understood. However, this has not been very clear for analytics models that possess both product and decision-making characteristics. On the product perspective, each analytics model can be seen as a result of production of chunks of data; data is the material to construct the analytics model in a certain

[27] KNIME, https://www.knime.com/

kind of form. On the decision-making perspective, an analytics model takes data as inputs and generates decisions. A data object itself is also an analytics model that directly outputs the input as a decision to the next analytics models. Both perspectives suggest data must be a constituent of an analytics model.

Standardization of interfaces is a key step leading to modularization. Standardization efforts of predictive analytics have been seen in the business intelligence and data analytics communities. The PMML is a well-adopted standard for formal representations of predictive models to facilitate exchanging them among data analytics tools. The DMN is another standard employed in development of decision models embedded in business decision processes. We employed these standards to differentiate the standard analytics models from the ad-hoc analytics models, as well as to capture the requirement decomposition and the structure decomposition of analytics models.

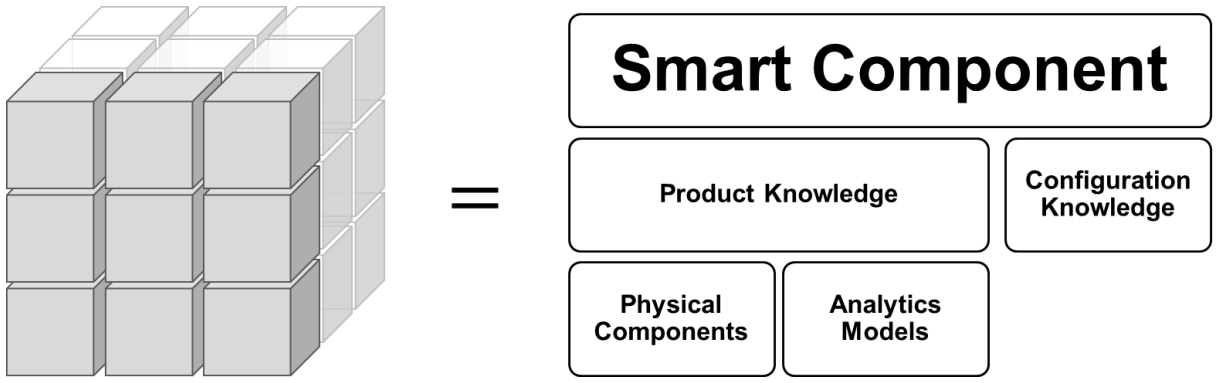
Two product family design approaches, scale-based and module-based approaches, had been adapted to develop parametric and combinatorial analytics models. A modularity function had been derived to evaluate the degree of modularity of an analytics model architecture. These methods were then applied to guide our development of the concept of analytics model lifecycle management, which had been implemented on top of an open-source PLM platform. We believe this provides a feasible solution to the A-T Space problem.

By providing capability for composition of datasets and heterogeneous analytics models in PLM, we also demonstrated how (1) to overcome the limitation of the present PMML standard that does not support the data representation, and (2) to reuse a unit model that has been encapsulated into a single PMML document with other unit models. The proposed technique in implementing the analytics model in PLM could provide a reference for future extension or enhancement of the current PMML specification. The lifecycle and workflow models available in the PLM system are leveraged to track

and trace both unit and composite models in various use case scenarios. This makes it easier to update, replace, and maintain unit predictive models.

Chapter 4

P-A Space: *Smart Component (sComponent) – Modeling Interoperability*



Now, we move to the P-A space of the Smart Products Hypercube. By our definition, a smart component shall be composed of at least one physical component for its body implementation and at least one analytics model for its intelligence implementation, as well as a set of configuration rules to reflect their interface relationships. As discussed previously, the information standards for physical components and analytics models have been modelled differently, so the first question would be: can we unify them by a higher-level abstraction so that we can maximally utilizing the existing efforts. If we can unify them, then how can we connect and compose them as a modular, unit component that can be reused, traced, maintained, and replaced on demand?

4.1 Smart Component Modeling

4.1.1 Comparison between Physical Component and Analytics Model

The concept established in the previous chapter provides a comparable way to analogize the

lifecycle of data analytics with that of physical product development. An analytics model is the product of the data analytics process. An analytics model is also an information-processing unit to produce decisions.

Table 4.1 Comparison between a Physical Component and an Analytics Model (Li et al., 2015b)

	Physical Component	***Analytics Model (e.g. Predictive Model)***
End Product	An end-product of manufacturing processes	An end-product of computational processes
Development Process	Integrated product design and manufacturing process	A formal knowledge discovery and data mining (KDDM) process
Process Planning	Output: • Manufacturing process selection • Material selection • Operation sequence Resource: • Raw material, work-in-progress stock • Fixture • Machine/tool • Operations (milling, turning, drilling, …)	Output: • Attribute selection • Analytical model selection • Step sequence Resource: • Raw data, intermediate data • Data extract, transform, load (ETL), data pre/post-processing • Algorithms (regression, classification, clustering, association Rules…)
Authoring and Management Tools	• Computer Aided Design/Manufacturing (CAD/CAM) • Product Data Management (PDM) • Product Lifecycle Management (PLM)	• Mathematics/Statistics tools • Data Mining packages • Application Lifecycle Management (ALM)
Visualization	2D drawings/3D models	2D/3D plots
Standards/Guidelines	ISO 10303 STEP, ISO 14306 JT, IGES (Initial Graphics Exchange Specification), …	PMML, CRISP-DM, PFA (Portable Format for Analytics), …

Table 4.1 lists a side-by-side comparison between a physical component and a predictive model, revealing that they share commonalities in many aspects, e.g., *authoring and management*, *production planning*, *visualization*, and *standards*. Both are produced by certain producers and are used by one or more consumers, to fulfill certain designed functions. Individual component/model is produced from certain kinds of raw materials or raw data, and could be supplied by various vendors. The production of the individual consumes resources and it needs deliberate production planning. A master model (the final physical product or analytics model) typically consists of a set of sub models. Each sub model may be composed of several unit models. Thus, the master model can be represented as an assembly tree. Models with common functions can be modularized and standardized for easier reuse, interchange, and composition. A model might be used repeatedly in the same master model or in a different master model

and may have variations, thus the model utilization history needs be traced over its lifecycle stages.

This analogy leads to a unified "Smart Component" abstract model shown as in Figure 4.1. Typically, a physical component model consists of metadata definition (e.g., attributes), material definition, structure (i.e., Bill of Materials), alternative components, manufacturer information, relevant documents (e.g., requirement specification and maintenance manual), and neutral electronic files (e.g. ISO 10303 STEP file) for long-term storage and data exchange. Similarly, an analytics model also contains these kinds of information. The main difference is that the dataset is treated as the primary "material" for the analytics model.

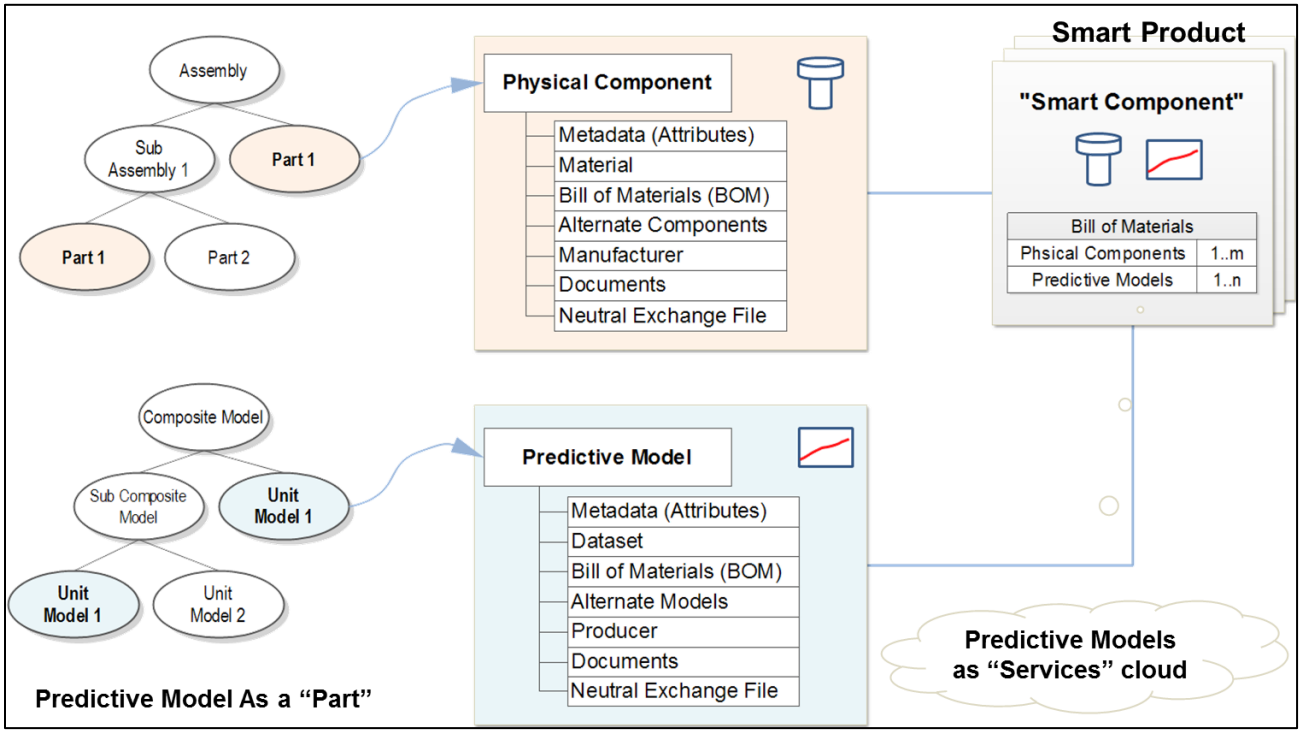


Figure 4.1 A Smart Component data model (Li et al., 2015b)

An analytics model (e.g. the predictive model in Figure 4.1) can be treated either as a "part" of a product or as a "service" for the product, depending on its purpose and location. Once physical components and analytics models are modeled in a unified way, they can be composed according to

certain configuration rules. Since PLM had been applied to manage both information of physical products (traditionally) and data products (as we just did previously), now it is also possible to be applied to manage the lifecycle information of smart products. The key is how we implement this abstraction in the PLM.

4.2.2 Product Shell and Smart Component Shell

The I4.0 Component defined in the RAMI4.0 framework provides a unified model for description of assets used in a manufacturing plant. An I4.0 component can be a production system, an individual machine or station, or an assembly inside a machine. It consists of an asset enriched by a so-called *Administration Shell* that contains the metadata of the real asset, status data of the asset, and all data generated during the asset lifecycle.

As shown previously (Figure 2.4, Chapter 2), the administration shell contains the data for *Virtual Representation* and the functions of the *Technical Functionality* (Adolphs et al. 2015). The *Manifest*, as part of the virtual representation, describes the necessary administrative details about the I4.0 component. The *Resource Manager* is also part of the administration shell, with which IT services have access to the data and functions of the administration shell and make them externally available. The administration shell and its contents can be hosted within one of the objects of an embedded system (the *active* mode) or distributed among one or more higher lever IT systems (the *passive* mode).

The administration shell provides a unified way to translate an arbitrary object (or a thing) to an intelligent object. However, the administration shell has to be implemented differently involving real-world scenarios. Thought of the product perspective of an analytics model, it can be encapsulated into a *Product Shell* to become a product without geometric form, instead, it has a mathematical representation. Then, the data model standards established in manufacturing domain (e.g. the ISO STEP AP242) can be readily applied, and the two types of products (physical product and data product) can be

composed using the approaches for product configuration, see Figure 4.2 and 4.3.

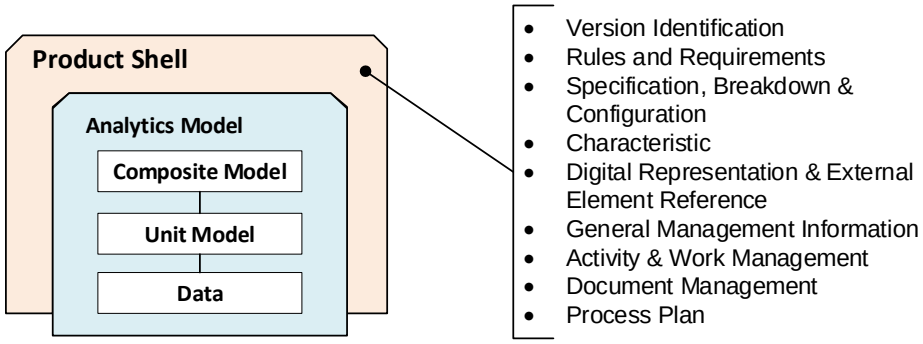


Figure 4.2 Product Shell for analytics models

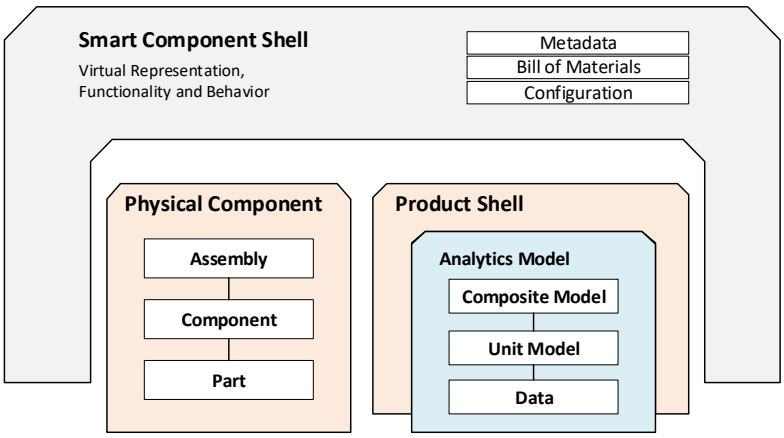


Figure 4.3 Smart Component Shell for unification of physical components and analytics models

A smart component (sComponent) can be seen as a special case of an I4.0 component in the context of smart products development. A notable difference of the sComponent from the I4.0 Component is we treat the analytics models as products rather than see them as functions in the administration shell. Thought of this way, the Smart Component concept discussed in this chapter contributes in two perspectives: (1) it provides an approach to implement the product layer of the RAMI4.0 hierarchy levels axis (Figure 2.3, Chapter 2); and (2) the capability to upgrade a smart component to an I4.0 Component implies the information defined for the sComponent can be compliant with the RAMI4.0 architecture so that the smart component data model can be easily extended for “smart services” and be applied in boarder contexts in the smart manufacturing field.

4.1.3 Information Dependency between Physical Components and Analytics Models

Let's recall the smart product schema depicted in Figure 1.9 (Chapter 1). A smart product perceives the environment using sensors, then processes the collected data/information for situational awareness as well as learning new knowledge. The existing knowledge and the emerging knowledge are collectively used to process the newly collected data for cognition. Then, the product can take actions to interact with its users or impact the environment through its actuators. This process can be abstracted as a *Sensor-Data-Model-Decision-Actuator* (SDMDA) unit model. The data fed into the analytics model and the decisions output from the model provide the bridge (or interface) that connects physical components and analytics models (Figure 4.4).

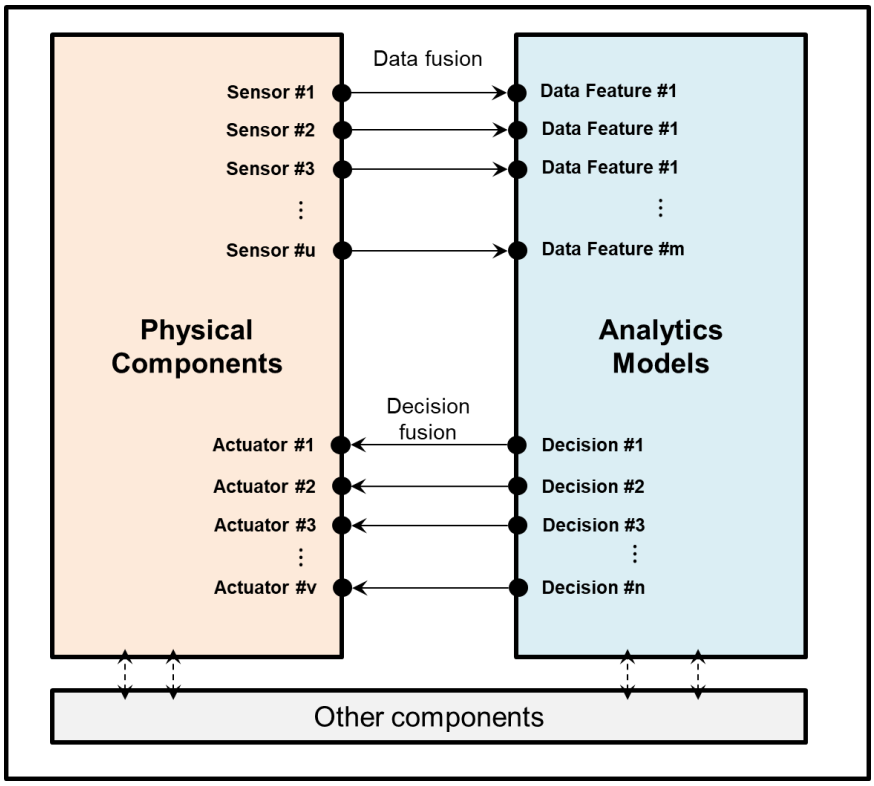


Figure 4.4 Interfaces between physical components and analytics models

Since data can from multiple sensors and each sensor may be related to sensing multiple data points, this often involves a data fusion process. Similarly, decisions can go to multiple actuators and each

actuator may be related to multiple decision actions, which often involves decision fusion process. Let's define the number of sensors, the number of actuators, the number of data features required by the analytics model, and the number of decisions output from the analytics model as u, v, m, and n, respectively. The number of incoming links to the analytics model is in the range [max{u,m}, uxm], and the number of outgoing links from the analytics model is in the range [max{v,n}, vxn].

Based on Durrant-Whyte (1988), the data from different sensors can be (1) *complementary*, when the information provided by the input sources represents different parts of the scene and could thus be used to obtain more complete global information; (2) *redundant*, when two or more input sources provide information about the same target and could thus be fused to increment the confidence; and (3) *cooperative*, when the provided information is combined into new information that is typically more complex than the original ones. Furthermore, the inputs and outputs of a data fusion system can be any of the five patterns (Dasarathy, 1997): (1) *data in-data out* (DAI-DAO); (2) *data in-feature out* (DAI-FEO); (3) *feature in-feature out* (FEI-FEO); (4) *feature in-decision out* (FEI-DEO); and (5) *decision in-decision out* (DEI-DEO). The data/decision fusion perspective is consistent with our decision-making perspective of analytics models discussed previously.

4.1.3 Product Configuration across Domains

Configuration of a smart product requires appropriate combination of physical parts, software, and services. A *configurable product (product platform)* is composed of components that are connected via ports to form a hierarchical structure (Soininen et al., 1998). The configuration of a physical product can be done at different positions along the supply value chain: producer, logistics, retailer, and customer (Schenk and Seelmann-Eggebert, 2003). A *configurable software (software product line)* is a set of software sharing a common set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribe way (Clements and

Northrop, 2001). A *configurable service* represents a service that is customized from a set of predefined options in order to fit the needs of individual customers (Böhmann et al., 2003). Inter-domain product configuration problems, e.g. configuration of software and product (Aurich et al., 2009), configuration of product-service system (Hubaux et al., 2012), have been reported but literatures are quite few.

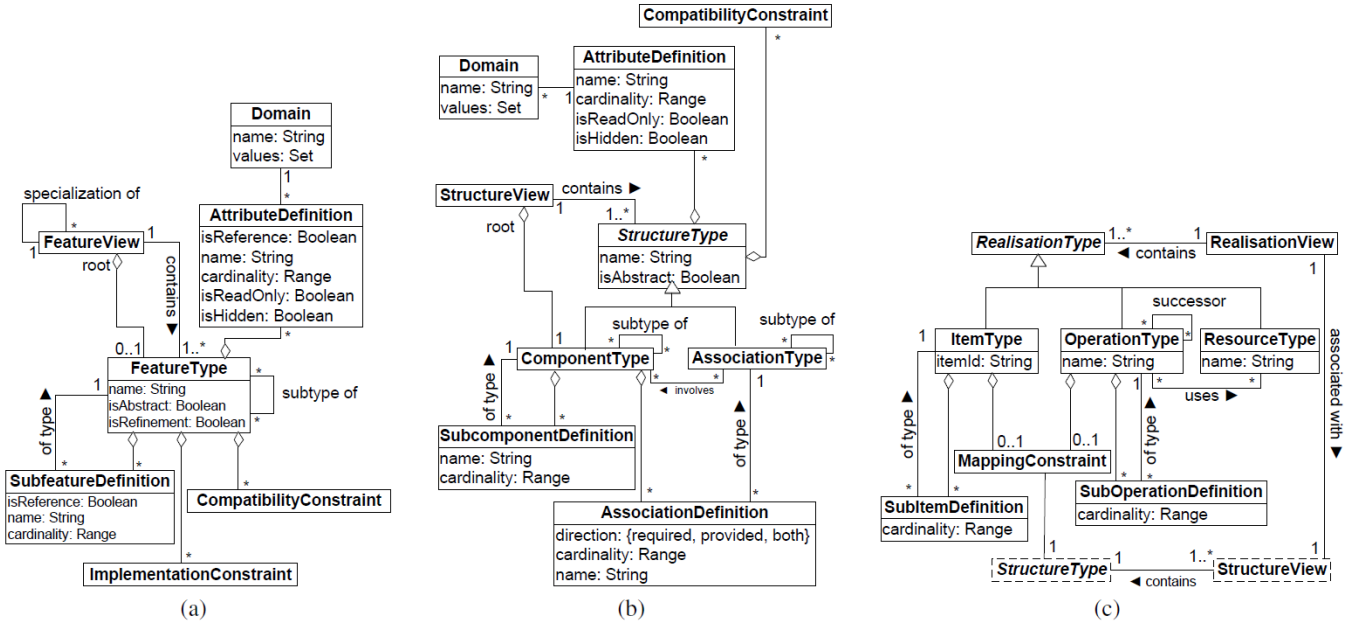


Figure 4.5 Product configuration across domains (Quéva et al., 2011): (a) Feature view (b) Structure view (c) Realization view

Quéva et al. (2011) proposed a framework for across domain product configuration using three different types of views: *Feature*, *Structure*, and *Realization*. While this work only discussed how to configure a product across physical, software, and service domains, its model-view abstraction provides a way to incorporate data analytics models either as a part or a service. More specifically, the core elements of the framework are defined as below:

- *Feature views* provide a set of views of a product family from a high level of abstraction. Product individuals can be characterized by the features (or functions) they provide. A feature view is composed of *feature types*. Variability is defined in each feature type using attributes that can take different values. Two types of constraints can be added to each type. *Compatibility constraints* model dependencies between the feature views, i.e. it specifies conditions that must hold in a valid configuration.

Implementation constraints model the dependencies between different types of views.

- *Structure views* define the different design components that realize the described features of the product family, and the relations between them. The compositional structure of the product families is often used to represent the structure data knowledge. A structure view is composed of *structure types* that can be either component types or association types. A physical structure view represents the physical structure of the product family. Component types are entities whose individuals are physical components involved in the physical design, while association types are used to model non-directional physical links between two components. A software structure view describes the architecture of the software system involved in the product family. Instances of component types represent software components, and association can be defined to model interfaces. A service structure view describes the specifications for the service to be delivered. Component types are service element types, and describes contractual agreements of what to be delivered.
- *Realization views* offer a detailed technical view of how the product individuals are realized. Realization view are aimed at describing the elements necessary for the concrete realization of the system for that dimension. The building blocks of a realization view are realization types: *item types*, *operations types*, and *resource types*. Item types represent the production components used to realize the products. It can be a BOM item for manufactured parts, a software package, or an object to be produced to deliver a service. Operation types are used to specify a set of operations needed during the production of individuals. Resource types may describe a machine, an operator, an information or anything that may be necessary to complete the operations.
- *Compatibility Constraints*: a compatibility constraint is specific to a particular view, and can only involve properties of this view. The evaluation of a constraint occurs during configuration, when types are instantiated to individuals.
- *Implementation Constraints*: Implementation constraints model the interaction between the base feature view and the structure views.
- *Mapping constraints* are defined in realization views to specify under which conditions a realization type should be included in the configuration results. Mapping constraints are declared in item and operation types, and refers to attributes from the structural type defined as context.

Since the three views are not domain specific, they can be extended to support data analytics models based on our concepts established in Chapter 3. Let's use the CNC energy consumption predictive model (Section 3.3.1) to illustrate:

- Feature view level: The purpose of this example predictive model is to predict the energy consumption of a given CNC machining center, so this is the feature type. The attributes are the process parameters used to define the predictive model.
- Structure view level: The energy consumption predictive model has its architecture which can be represented as a model structure view, and its individual types of analytics models (regression or neural network model) can be represented by corresponding component type. The compatibility constraints can be established between the data dictionary required by the individual analytics model and the data fields present in the relevant datasets, for example, $Prediction\ of\ TotalPower \Rightarrow existence\ of\ FeedRate, SpindleSpeed, CuttingDepth, and\ CuttingDiameter$.
- Realization view level: The energy prediction model can be represented conforming with the PMML specification and can be implemented as a digital model in PLM.

4.2 Smart Component Implementation

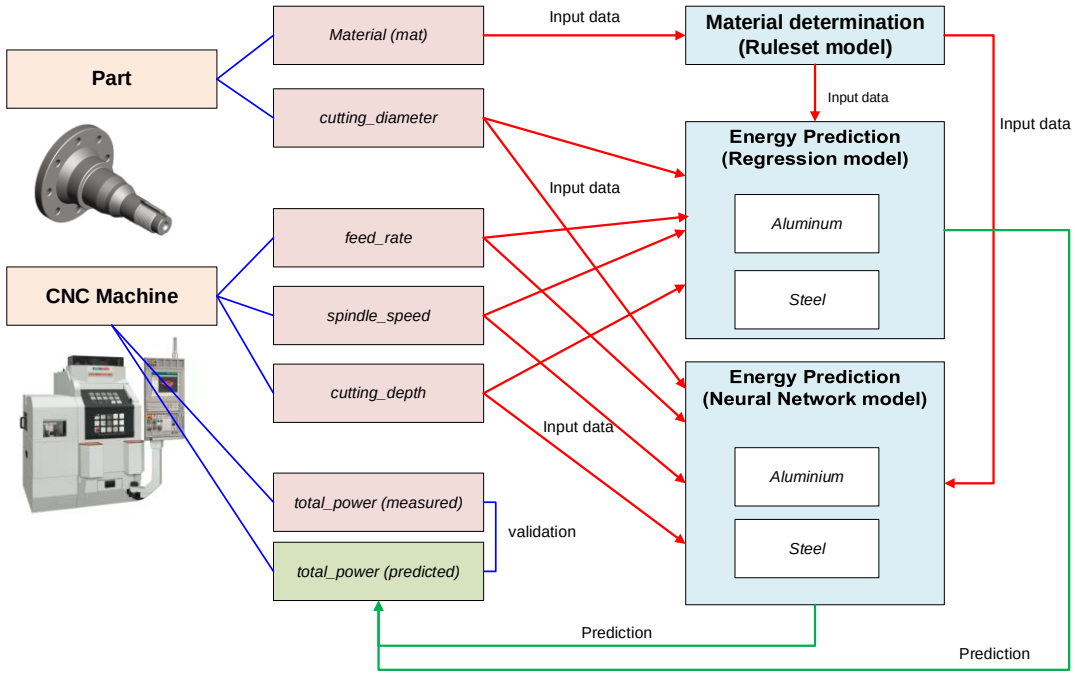


Figure 4.6 Data flow of the CNC machining center energy prediction

Now, let's discuss how to implement the abovementioned concepts in PLM. To illustrate, let's revisit the CNC machining center case. A CNC machining center can be equipped with an energy-

consumption predictive model to estimate its power usage based on process parameters, which include current feed rate, spindle speed, cutting depth, and cutting diameter (Figure 4.6). The values for these parameters are collected from the machine's built-in sensors. In this case, the energy consumption predictive model can be seen as a part of the CNC machining center. On the other hand, when we consider the part produced by this machine, the energy consumption predictive model becomes a service for the part because the predictive model functions outside the part, which is depicted in Figure 4.7.

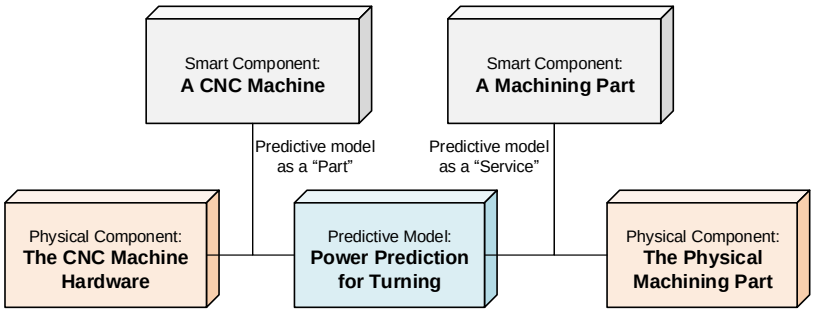


Figure 4.7 Predictive model as a part (to the CNC machining center) or a service (to the part) (Li et al., 2015b)

4.2.1 Smart Component ItemType and Configuration

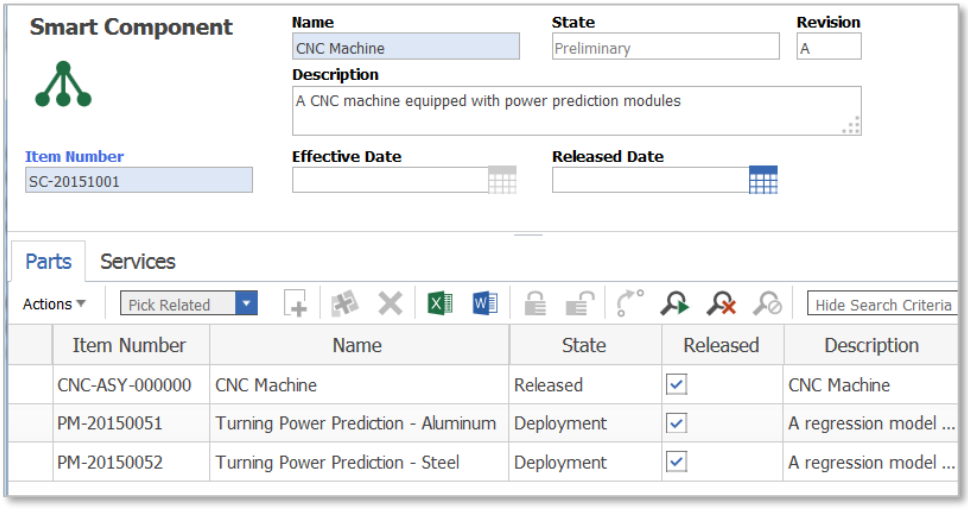


Figure 4.8 The smart component implementation in PLM (Li et al., 2015b)

The *Smart Component Shell* concept can be implemented as a hybrid item in PLM. It defines the common properties for heterogeneous item types. We apply different relationships to connect the

physical components to “part” predictive models and to “service” predictive models (Figure 4.8). The part relationship requires a strict composition relationship between the smart component and the analytics model. The service relationship is a weaker association relationship and it is optional to a smart component. Similar to the configuration rules for composition of analytics models, the configuration rules for composition of physical-analytics models can also be represented as a ruleset model.

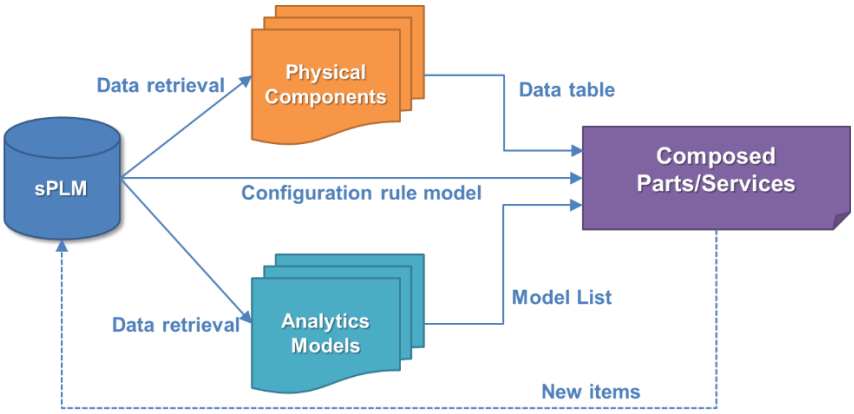


Figure 4.9 Smart Component model composition

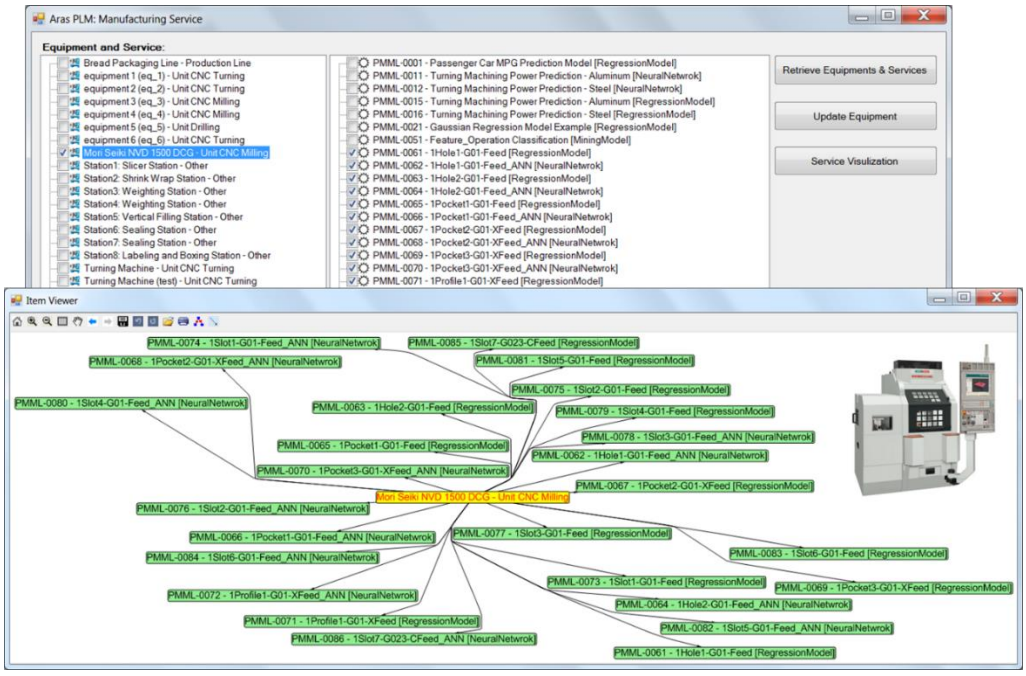


Figure 4.10 Composition of physical components and analytics models for the CNC machining center.

Figure 4.9 and 4.10 demonstrate that when a configuration engineer is working on the physical

components, the available, compatible analytics models should be dynamically filtered out and be recommended based on the configuration contexts and rules. The configuration information will be stored back to the PLM system as a bill of materials or a bill of services for history tracing. In addition, the configuration can be graphically visualized.

4.2.2 BOM, Product Family, and ECN – A Collaboration Scenario

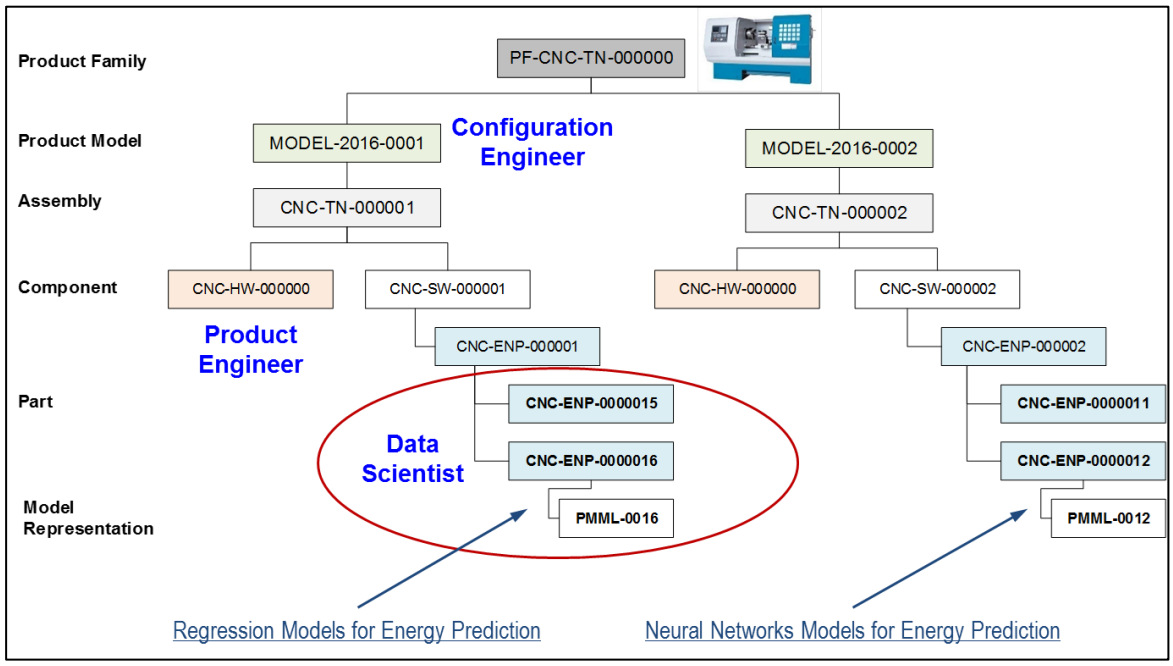


Figure 4.11 Two instances of a CNC machining center product family

Now, let's illustrate how the smart component data model does facilitate the collaboration between product engineers, data scientists, and configuration engineers. Assume there are two instances of the CNC machining center product family. One instance is equipped with a power consumption module of a polynomial regression model, while the other instance can be equipped with a power consumption module of a neural network model, see Figure 4.11. A product engineer, a data scientist, and a configuration engineer are involved in the product development. The product engineer is primarily responsible for physical product design, the data scientist is responsible for developing data-analytics

models for the intelligence parts of the product, and the configuration engineer is responsible for product configuration.

Figure 4.12 shows the data flow for this scenario. The workflow is as follows: (1) the product engineer (PE-1) submits and releases the physical product design of the CNC machining center; (2) the data scientist (DE-1) develops and submits an energy prediction analytics model (linear regression model) for the CNC machining center; (3) the configuration engineer (CE-1) creates a new product configuration incorporating the physical product design and the predictive model; (4) the data scientist (DS-1) then updates the energy prediction model with a polynomial regression model, an Engineering Change Notice (ECN) is generated to track this change impact; (5) the configuration engineer (CE-1) then updates the product configuration to a new version. During the process, all relevant documents are recorded and all involved items and activities can be traced and visualized.

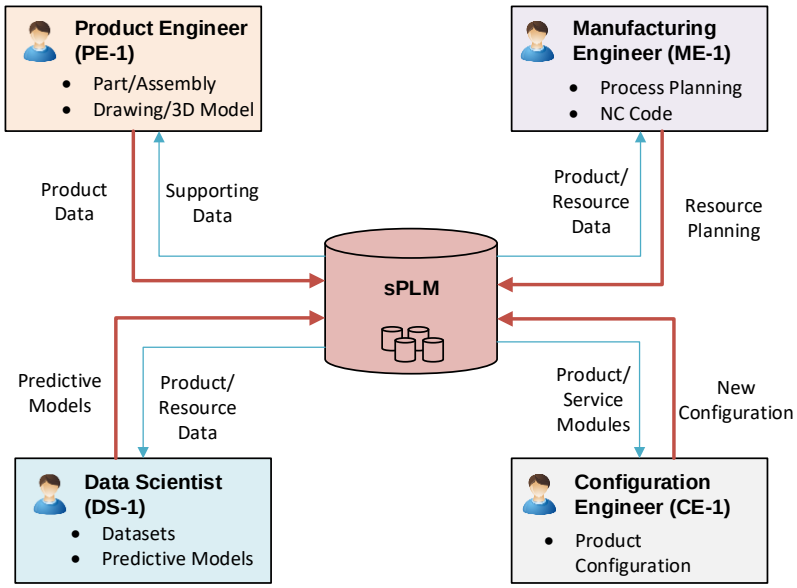


Figure 4.12 The data flow of a multidisciplinary collaboration scenario

Figure 4.13 shows the bill of materials including physical components as well as analytics models. Figure 4.14 shows the engineering change history that is controlled by the ECN workflow. We can also compare the different versions of the product configurations.

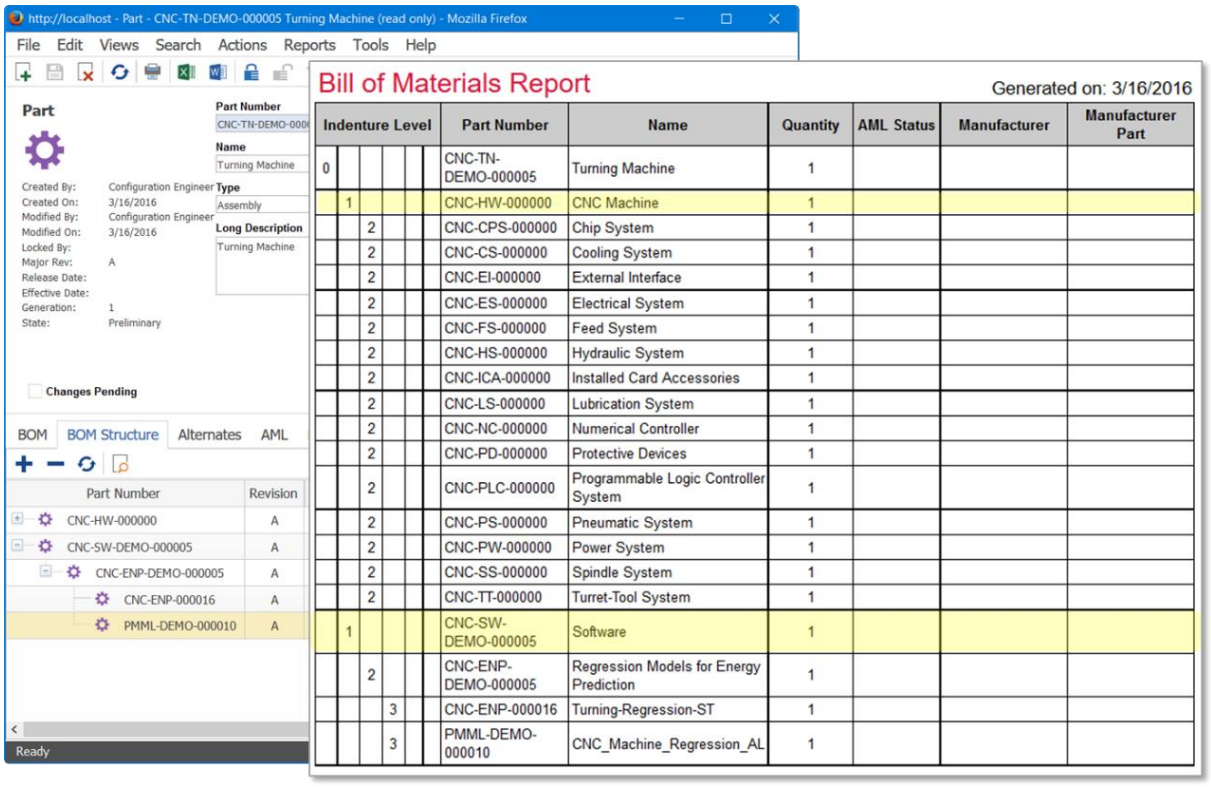


Figure 4.13 BOM of the CNC machining center with physical component, software, and analytics models

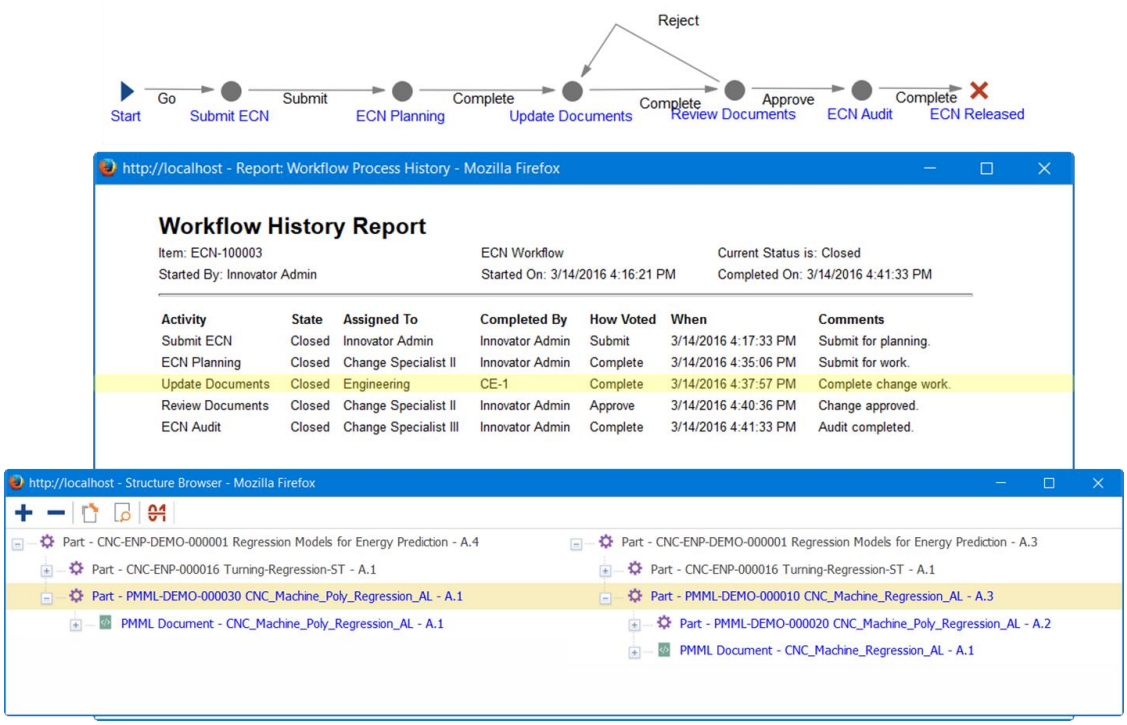


Figure 4.14 Engineering change history and versioning of the CNC machining center

4.2.3 Smart Component Retrieval and Execution

Both the CNC machining center and the part can also use other analytics models as services; for instance, a process planning model that defines the operation sequence to produce machined parts. In our previous work (Li et al., 2015a), we illustrated a model to predict manufacturing operations for geometric features of a prismatic part. That model is comprised of two unit-models: a ruleset model and a tree model. The ruleset model is for non-hole features such as a face, a slot, or a pocket; the tree model is for hole features. These two unit-models can be combined together to make prediction for different feature categories. If the CNC machining center is equipped with this process planning model, it will be aware of the CNC machining center as a resource and will take consideration of it with other constraints when any process planning process is initiated.

Putting pieces together, Figure 4.15 shows the different levels of composition to form the smart CNC machining center with multiple types of analytics features.

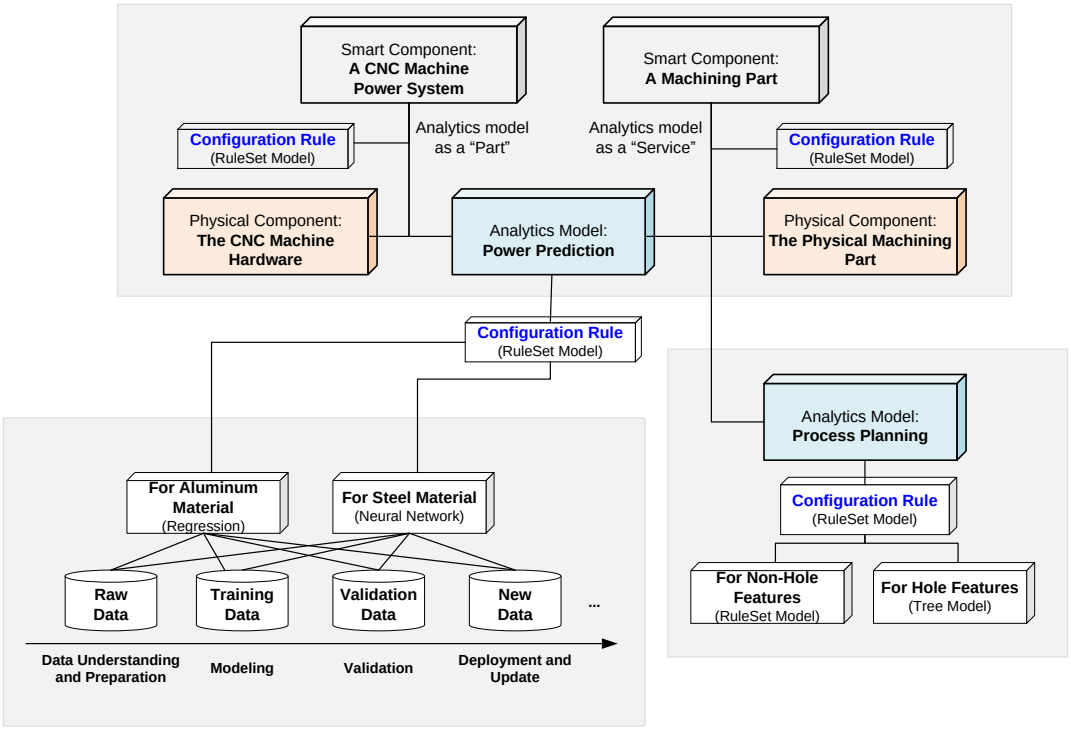


Figure 4.15 Different levels of composition to form the smart CNC machining center

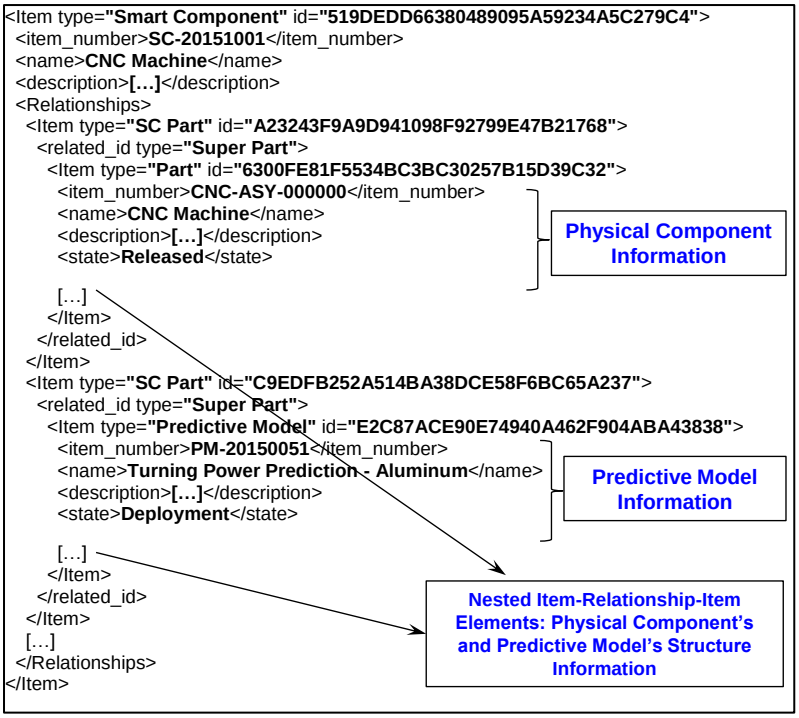


Figure 4.16 The XML-based response from the PLM system (Li et al., 2015b)

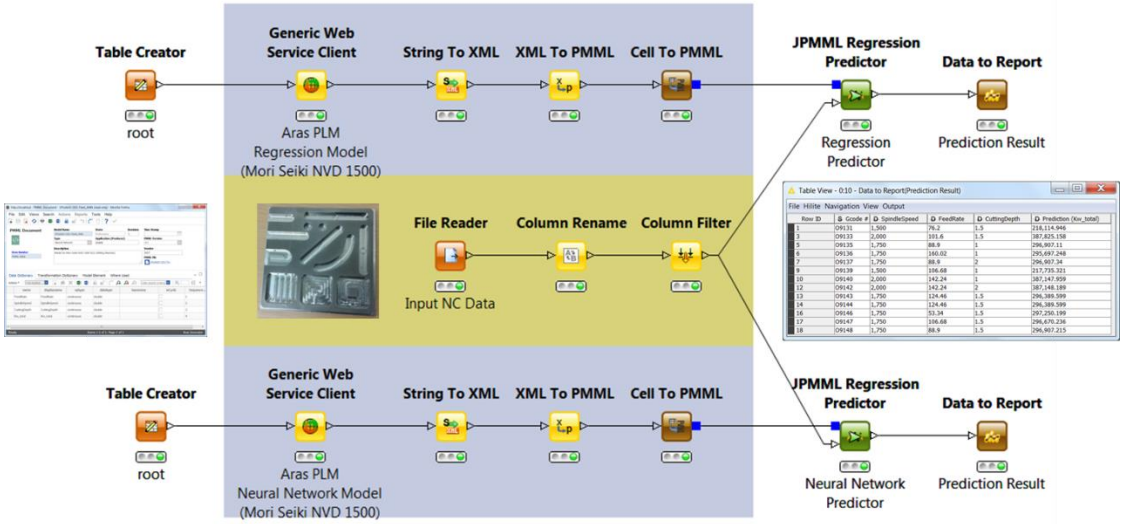


Figure 4.17 Energy consumption prediction for a CNC machining center

The composite smart product model can then be retrieved at any level from inside PLM and/or by outside applications. Figure 4.16 shows a skeleton XML data from the PLM system that responds to a model query. The response to the query includes retrieval of (1) all the necessary model data such as

physical components, predictive models and their relevant datasets, and (2) all lifecycle information such as lifecycle stages and revisions. These data can be further parsed by the rule engine and the scoring engine embedded inside the PLM system or used by an external application. Figure 4.17 demonstrates how an KNIME user can retrieve the CNC machining center's feature data and its associated energy prediction models, and then consume the data and models in KNIME.

4.3 Case Study: *A Modular Framework for Sustainability Assessment*

In this section, we demonstrate the unification of physical product information and data analytics model information is not only beneficial to smart products development, the concept can also be applied to broader applications in smart, sustainable manufacturing.

Life cycle assessment (LCA) has been an effective approach for sustainability assessment of a product and its related processes. It involves collecting inventory data and modeling processes related to each of the product lifecycle stages and in various abstract levels. Oftentimes, the lack of understandings of the physical processes makes it challenging to achieve a reliable sustainability assessment. Data-driven modeling has emerged as a complementary approach that takes advantage of machine learning techniques to embellish the current practices based on expert knowledge. Below we briefly present a modular, hybrid framework where both the top-down knowledge-based models and bottom-up data-driven models can be accommodated for and combined at different abstract levels in assessing sustainability. More details can be seen in Li et al. (2017b).

4.3.1 A modular hybrid LCA framework

As shown in Figure 4.18, an LCA procedure can be decomposed along two dimensions: the *product lifecycle stages* and the *LCA decision-making activities*. The horizontal axis represents the material flow along the product lifecycle that begins with the raw materials gathering and ends at the point when all

materials are recycled or treated as wastes. The dashed line indicates the lifecycle could be closed-loop. Each lifecycle stage involves a hierarchy of processes and activities. For instance, a product can include parts produced by different manufacturing methods (e.g., machining or injection molding).

The vertical axis represents the LCA decision-making system that takes product lifecycle data (geometry, material, production operations, maintenance, etc.) to reach multi-level LCA analyses. The scoring can be done for any individual activity, process, stage, or any aggregations.

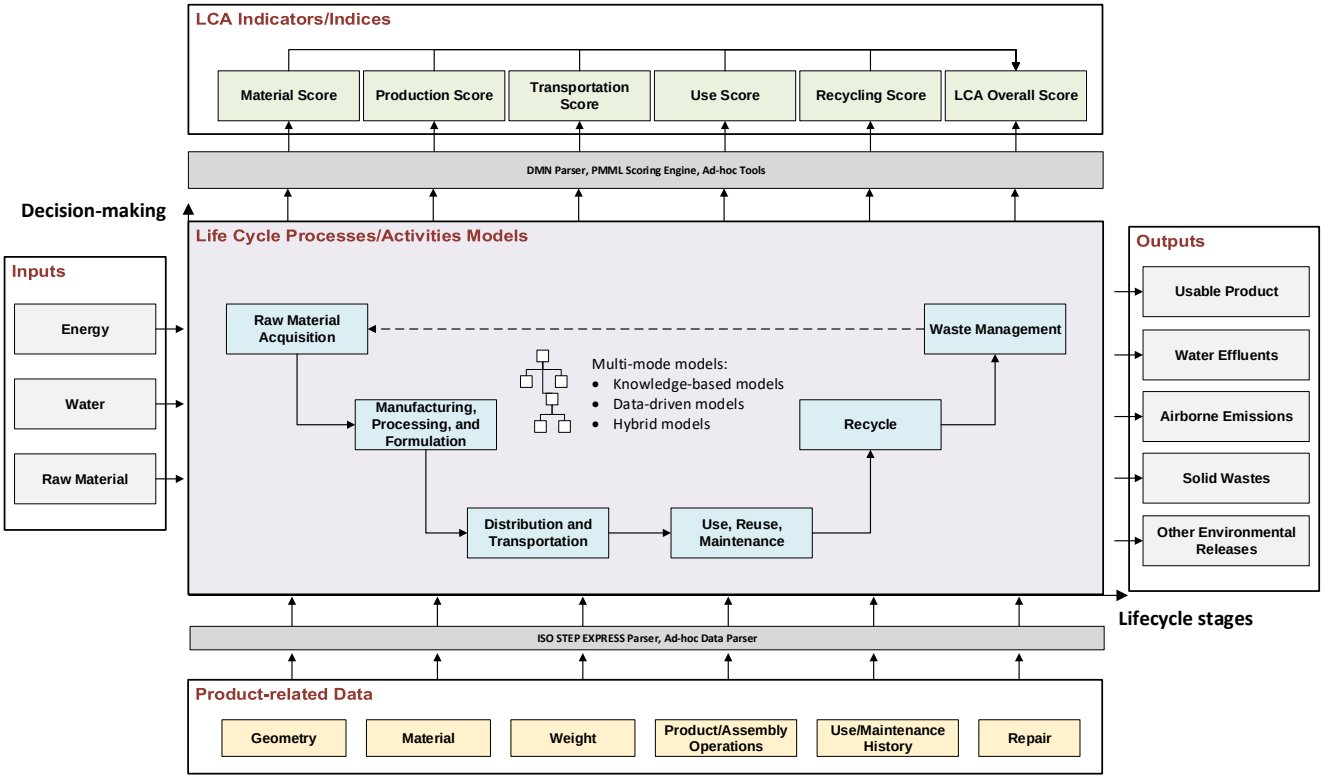


Figure 4.18 A modular, hybrid LCA framework (Li et al., 2017b)

The modular design of an analytics model architecture involves model structure (from the product perspective) and decision logic (from the decision perspective) decompositions. We employ PMML and DMN to establish this modular LCA framework that accommodates knowledge-based, data-driven, and hybrid models. The PMML standard is used to represent analytics models that have the standardized forms. For non-standard models, their original forms can be used and ad-hoc tools for interpreting and

executing them can be invoked. The DMN standard is used to connect data, models, and the resultant decisions to form a decision network. We also use the ISO STEP standard and its extensions to represent and exchange the product lifecycle data in an LCA context (Li and Roy, 2014). The parsers for ISO STEP, DMN, and PMML are to interpret the data/model encoded in these standardized formats.

4.3.2 Implementation Using Eco-Indicator'99 LCA Model

The framework is implemented based on the Eco-Indicator'99 LCA formulation and is verified using a stapler product (consisting of 14 parts). The top-level of the LCA model is encoded as an *ensemble* of *Scorecard* models (Figure 4.19).

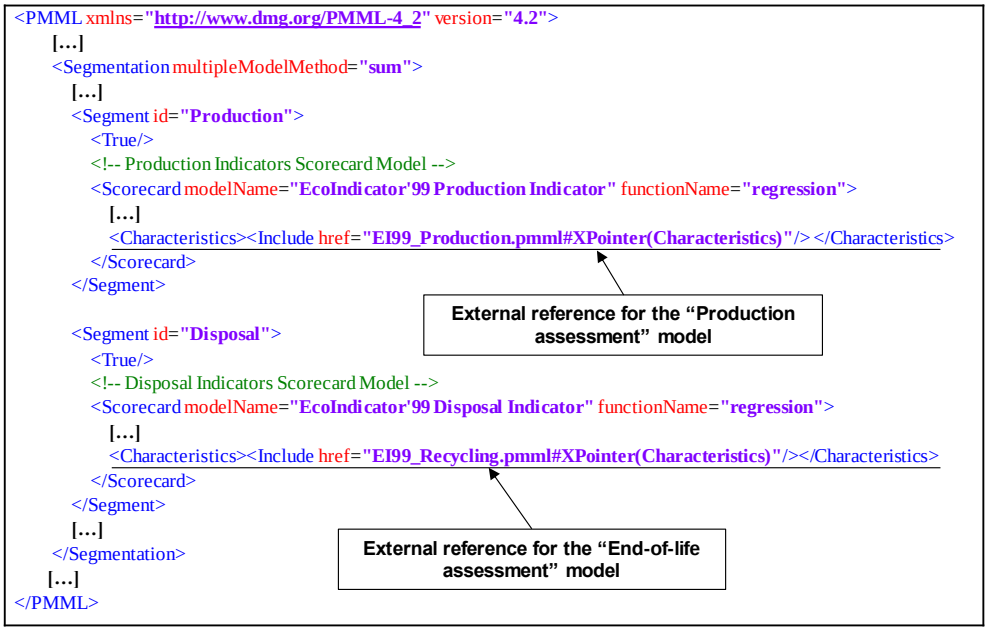


Figure 4.19 A skeleton LCA model in PMML: ensemble of Scorecard models (Li et al., 2017b)

It is noted the current PMML specification provides no support for external document references; consequently, there are reductions in the flexibility of exchange, reuse, replacement, versioning, and tracing of the individual models. For example, there are scenarios where two or more models share unit or component models. A simple remedial solution is to use the *XML/Include* and *XML/XPointer*

elements to link a master PMML model to external PMML models, since the PMML is conforming to the XML (Extensible Markup Language) specification.

XInclude defines an inclusion mechanism that facilitates modularity in XML documents. The inclusion process is formally defined as merging a number of XML information sets into a single composite XML. *XPointer* stands for XML Pointer Language, an extensible system for addressing portions of an XML document. The skeleton LCA model shown in Figure 4.19 also demonstrates how we implemented using this technique. The two segments ("Production" and "Disposal") in this model each refers to an external Scorecard model.

```
<PMML xmlns="http://www.dmg.org/PMML-4_2" version="4.2">
    [...]

    <Scorecard modelName="EcoIndicator'99 Production Indicator" functionName="regression">
        <MiningSchema>
            <MiningField name="WEIGHT" usageType="active" invalidValueTreatment="asMissing"/>
            <MiningField name="PRODUCTION_PROCESS" usageType="active" invalidValueTreatment="asMissing"/>
            <MiningField name="PRODUCTION_SCORE" usageType="target"/>
        </MiningSchema>
        <Output>
            <OutputField name="FINAL_SCORE" feature="predictedValue" optype="continuous" dataType="double"/>
            <OutputField name="REASON_CODE" feature="reasonCode" optype="categorical" dataType="string"/>
        </Output>
        <Characteristics>
            <Characteristic name="productionIndicator" reasonCode="Production Indicator" baselineScore="0">
                [...]
                <Attribute partialScore="21" reasonCode="per kg PE, PP, PS, ABS, without production of material">
                    <SimplePredicate field="PRODUCTION_PROCESS" operator="equal" value="INJECTION MOULDING - 1"/>
                    <ComplexPartialScore>
                        <Apply function="*">
                            <Constant>21</Constant>
                            <FieldRef field="WEIGHT"/>
                        </Apply>
                    </ComplexPartialScore>
                </Attribute>
                [...]
                <!-- Default True Statement -->
                <Attribute partialScore="0">
                    <True/>
                </Attribute>
            </Characteristic>
        </Characteristics>
    </Scorecard>
</PMML>
```

Input data

Output score

A predefined indicator or an embedded model for a specific process

Figure 4.20 A Scorecard example for *injection molding* production (Li et al., 2017b)

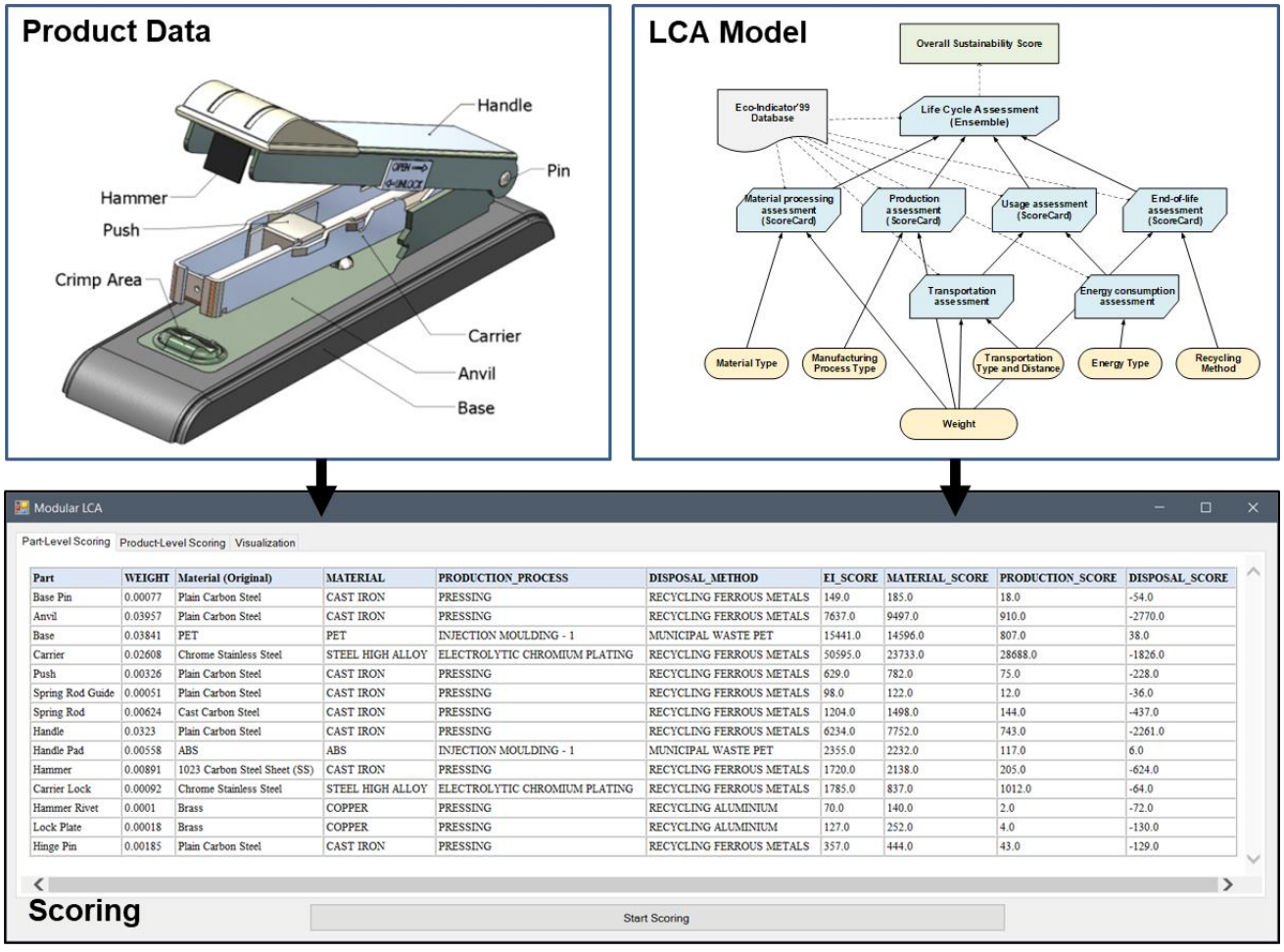
Each Scorecard model can be independently used for assessing the sustainability performance of a product lifecycle stage. For example, the *Production assessment* module shown in Figure 4.20 is a Scorecard model that takes the inputs including the *material weight* and the compatible *Manufacturing process type*, and outputting a *production score*.

Table 4.2 A segment of the STEP representation of the stapler's material and disposal method (Li and Roy, 2014)

```
#41001=PRODUCT('Recycling Ferrous Metals', 'Recycling Ferrous Metals', '', (#2));
#41002=PRODUCT_RELATED_PRODUCT_CATEGORY('Disposal Process', '', (#41001));
#41003=PRODUCT_DEFINITION_FORMATION('1', 'LAST_VERSION', #41001);
#41004=PRODUCT_DEFINITION_CONTEXT('process definition', #1, 'disposal');
#41005=PRODUCT_DEFINITION('process definition', '', #41003, #41004);
…

#11001=PRODUCT('Chrome Stainless Steel', 'Steel', '', (#2));
#11002=PRODUCT_RELATED_PRODUCT_CATEGORY('raw material', '', (#11001));
#11003=PRODUCT_DEFINITION_FORMATION('1', 'LAST_VERSION', #11001);
#11004=PRODUCT_DEFINITION_CONTEXT('material definition', #1, 'material extraction');
#11005=PRODUCT_DEFINITION('material definition', '', #11003, #11004);
…

#11099=MAKE_FROM_USAGE_OPTION('process to dispose Chrome Stainless Steel', 'disposed by', '', #11005, #41005, 1.0, '', #19001);
…
```



Part	WEIGHT	Material (Original)	MATERIAL	PRODUCTION_PROCESS	DISPOSAL_METHOD	EI_SCORE	MATERIAL_SCORE	PRODUCTION_SCORE	DISPOSAL_SCORE
Base Pin	0.00077	Plain Carbon Steel	CAST IRON	PRESSING	RECYCLING FERROUS METALS	149.0	185.0	18.0	-54.0
Anvil	0.03957	Plain Carbon Steel	CAST IRON	PRESSING	RECYCLING FERROUS METALS	7637.0	9497.0	910.0	-2770.0
Base	0.03841	PET	PET	INJECTION MOULDING - 1	MUNICIPAL WASTE PET	15441.0	14596.0	807.0	38.0
Carrier	0.02608	Chrome Stainless Steel	STEEL HIGH ALLOY	ELECTROLYTIC CHROMIUM PLATING	RECYCLING FERROUS METALS	50595.0	23733.0	28688.0	-1826.0
Push	0.00326	Plain Carbon Steel	CAST IRON	PRESSING	RECYCLING FERROUS METALS	629.0	782.0	75.0	-228.0
Spring Rod Guide	0.00051	Plain Carbon Steel	CAST IRON	PRESSING	RECYCLING FERROUS METALS	98.0	122.0	12.0	-36.0
Spring Rod	0.00624	Cast Carbon Steel	CAST IRON	PRESSING	RECYCLING FERROUS METALS	1204.0	1498.0	144.0	-437.0
Handle	0.0323	Plain Carbon Steel	CAST IRON	PRESSING	RECYCLING FERROUS METALS	6234.0	7752.0	743.0	-2261.0
Handle Pad	0.00558	ABS	ABS	INJECTION MOULDING - 1	MUNICIPAL WASTE PET	2355.0	2232.0	117.0	6.0
Hammer	0.00891	1023 Carbon Steel Sheet (SS)	CAST IRON	PRESSING	RECYCLING FERROUS METALS	1720.0	2138.0	205.0	-624.0
Carrier Lock	0.00092	Chrome Stainless Steel	STEEL HIGH ALLOY	ELECTROLYTIC CHROMIUM PLATING	RECYCLING FERROUS METALS	1785.0	837.0	1012.0	-64.0
Hammer Rivet	0.0001	Brass	COPPER	PRESSING	RECYCLING ALUMINIUM	70.0	140.0	2.0	-72.0
Lock Plate	0.00018	Brass	COPPER	PRESSING	RECYCLING ALUMINIUM	127.0	252.0	4.0	-130.0
Hinge Pin	0.00185	Plain Carbon Steel	CAST IRON	PRESSING	RECYCLING FERROUS METALS	357.0	444.0	43.0	-129.0

Figure 4.21 LCA scoring results: part-level scoring and product-level scoring (adapted from Li and Roy, 2014; Li et al., 2017b)

Each module can be further decomposed and represented as a modular PMML model. For example,

the indicators of *machining* and *injection molding* processes in the *Production assessment* module (designed as a combinatorial model) can be generated from a *Regression* model and a *Bayesian Network* model, respectively (Figure 4.22). These two individual models are then composed by implementing appropriate configuration rules as a *Ruleset* model.

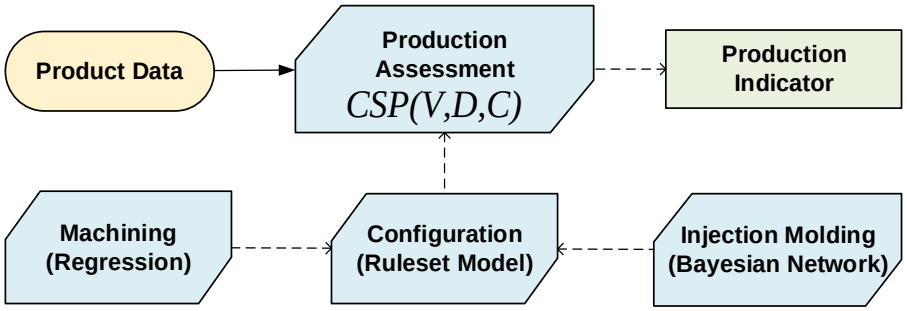


Figure 4.22 Modular production sustainability assessment (Li et al., 2017b)

In Li et al. (2017b), we established a Bayesian Network model to query for the energy consumption of the two injection molded parts (the Base and the Handle Pad) in the stapler product. The Eco-Indicator'99 predefines the energy indicator value of 21 million-points (a normalized value) for using injection molding to produce a part using one kg of PE, PP, ABS, or PS materials (without considering the production of the raw materials). To convert the energy consumption values in kJ to the Eco-Indicator'99 equivalent indicator, we assume that

$$Indicator = k * Average\left(E_{part}(PE), E_{part}(PP), E_{part}(ABS)\right), \quad (4.1)$$

where $Indicator$ is in million-points and E_{part} is in kJ. This results in $k = 4.09 \times 10^{-3}$ kJ^{-1}. Besides, this calculated indicator based on energy consumption values is comparable to the real Eco-Indicator, since the predefined injection molding indicator in Eco-Indicator'99 only considers energy consumption as well. For the staple's Base and Handle Pad parts, we also assume that the numbers of mold cavities are both 40. From querying the BN model and using the Equation 4.1, the calculated indicator obtained for the Base part is 4.5 and the indicator for the Handle Pad is 26.6. Compared to the

average value 21 for the injection molding process in Eco-Indicator'99, these two calculated indicators show the advantage of using the proposed modular, hybrid knowledge-based and data-driven LCA approach. The advantage is that the calculated indicator is that it is no longer an aggregated value; instead, it truly reflects the real energy consumption of the injection molding process given different product design, mold design, material selection, and process parameters.

4.4 Summary

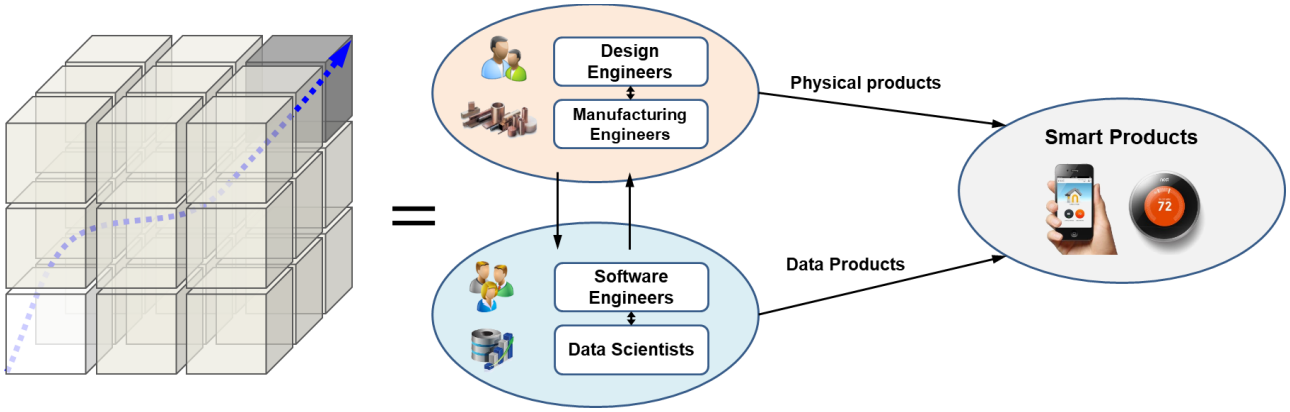
Based on the observed commonalities between physical components and analytics models, we derived a *Smart Component* data model to consistently compose both at the same time as a uniform, hybrid model. The data model enables engineers, data scientists, and other stakeholders to collaborate on a common PLM platform to develop smart products, e.g. a smart CNC machining center. The administration shell concept of I4.0 Component was used to embed product-augmented information for analytics models, and a *Smart Component Shell* was then used to connect physical components and productized analytics models. This way, a smart component can be seen as a special case of an I4.0 component in the context of smart products development. And it provides an approach to implement the product layer of the RAMI4.0. It also provides the possibility to extend the smart component data model for "smart services" and to apply it in boarder smart manufacturing field.

To validate, we applied the smart component concept to develop a modular, hybrid framework to accommodate both knowledge-based models and data-driven models at different abstract levels in assessing sustainability. Three well-adopted standards – the ISO STEP, DMN, and PMML were employed to capture the product-related data/information, to decompose the decision logics of the LCA-related analytics models, and to represent the individual analytics model structure. With the modular LCA framework, an Eco-Indicator'99-based LCA was implemented as an ensemble of Scorecard models

for individual lifecycle stages. Each sub-analytics model had been treated as an ensemble of lower-level models. The framework was implemented and verified using a stapler with two parts produced by injection molding processes.

Chapter 5

P-A-T Space: *NPD³ – Modeling Concept Development Process*



Up to now, we have discussed the feasibility to conceptualize a data analytics product so that we can uniformly elicit and compose knowledge to represent, exchange, and co-develop both physical and analytical components to construct a smart product. The next question is, how to decompose the tasks from the product development process for a dedicated data science team? In another words, what are the key tasks that the engineers in a physical product development team and the data scientists in a data products development team need to conduct? Accordingly, which tasks need to be coordinated across the two team groups? When and what information needs to be exchanged between the two groups to collectively achieve the product development? And more interestingly, what are the patterns and characteristics of their interactions?

To answer these questions, we revisit the existing process models for physical product development and data analytics, since each one has prescribed the common activities used in many practical projects. The standard steps and activities prescribed in these existing models provide an initial view of how engineers or data scientists individually work. We then hypothesize the potential collaboration points by aligning and comparing these models. This analysis helps to derive an initial integrated model that for

both engineers and data scientists. We then apply the hypothesized model to a real-world smart product development case. We develop an information decomposition framework to qualitatively categorize the observations of the interaction patterns within the case study, which leads us to achieve a theoretical framework for presenting the detailed interaction contents of the information flows. The interaction patterns and information contents complement our initial view of the integrated model that depicts the high-level key tasks and information flows.

To start, we focus on investigating the *Concept Development* stage of a new product development process. We believe an effective collaboration between engineers and data scientists in the front-end of the process will help avoid wasteful rework in the downstream processes and will enable the creation of better products that maximizes the potential of both the physical and analytical components of the product. Indeed, literature has reported that the time spent on the data understanding and preparation activities can take more than 60 percent of the overall data analytics efforts (Sapp, 2017).

5.1 NPD^3 Model: *An Integrated Process Model for New Product Development with Data-Driven Features*

Taking the information-processing perspective, the product development system becomes an information network. This product development information network usually consists of three levels of information-processing units (Distanont et al., 2012; Collins, et al., 2008): (1) the Overall Structure – the product development process as a whole is a single entity of tasks that share information; (2) the Subgroup – the groups of tasks that interact more with each other than with other tasks in the product development process; and (3) the *Individual Tasks* – the key tasks that are identified based on their relational roles as information transmitters (coordinator, gatekeeper, representative, liaison, or consultant).

To maximally leverage the existing models, we need to (1) identify the three levels of units already prescribed in the standard NPD model and the CRISP-DM model; (2) identify the prescribed tasks for individual roles. More specifically, in the NPD model we focus on the tasks prescribed for project manager, design engineer, and manufacturing engineer; in the CRISP-DM model we focus on the tasks prescribed for data scientists. Note that this dissertation employs the BPMN (Business Process Model and Notation) and DMN (Decision Model and Notation) conventions to represent process workflows and the decision-making logics in a data-driven product. Compared to other process diagramming approaches such as BPEL (Business Process Execution Language) and Petri Nets, BPMN focuses more on participants, and controls their interactions and flow with events and decisions (Debevoise and Taylor, 2014). BPMN and its companion, DMN, for modeling modular decision models, can be automated in a business process management system.

As mentioned previously, we focus on analyzing the concept development stage. The main engineering tasks prescribed in the NPD concept development stage include *Investigate feasibility of product concepts*, *Develop industrial design concepts*, *Build/test experimental prototypes*, *Estimate manufacturing cost*, and *Assess production feasibility* (Ulrich and Eppinger, 2012). Their activities and relevant data sources are shown in Table 5.1. Design engineers usually fulfill the first three tasks and manufacturing engineers typically fulfill the last two tasks.

Table 5.1 Concept development tasks, activities, and data sources (Ulrich and Eppinger 2012)

Task	Activitiy	Data Source
1. Identify Customer Needs	Gather new data from customers	• Interview • Focus groups • Observe the product in use
	Interpret the raw data in terms of customer needs	
	Organize the needs into a hierarchy of primary, secondary, and tertiary needs	
	Establish the relative importance of the needs	
	Reflect on the results and the process	
2. Establish target specification	Prepare the list of metrics	
	Collect competitive benchmarking information	

	Set ideal and marginally acceptable target value	
	Reflect on the results and the process	
3. Generate product concepts	Clarify the problem	• Understanding • Problem decomposition • Focus on critical subproblems
	Search externally	• Lead users • Experts • Patents • Literatures • benchmarking
	Search internally (brainstorming)	• Individual • Group
	Explore systematically	• Concept classification tree • Concept combination table
	Reflect on the solutions and the process	
4. Concept selection	Prepare the selection matrix	
	Rate the concepts	
	Rank the concepts	
	Combine and improve the concepts	
5. Concept testing	Define the purpose of the concept test	
	Choose a survey population	• Face-to-face interaction • Telephone • Mail/email • Internet
	Choose a survey format	
	Communicate the concept	
	Measure customer response	
	Interpret the results	
	Reflect on the results and the process	
6. Set final specification	Develop technical models of the product	▪ Simulation / prototype
	Develop a cost model of the product	▪ Bill of Materials (BOM)
	Refine the specifications, making trade-offs where necessary	▪ Optimization
	Flow down the specifications as appropriate	
	Reflect on the results and the process	

According to Marbán et al. (2009), the tasks defined in the CRISP-DM that are relevant to concept development (for data products) are mainly in the *Business Understanding* and *Data Understanding* stages. We argue that the concept development should focus on the role of translating business needs into technical implementation specifications. Therefore, we align the CRISP-DM's *Business Understanding* stage with the NPD's *Planning* stage, and we only count the tasks defined in the *Data Understanding* stage as concept development activities for data products. These tasks include: *Collect data*, *Describe data*, *Explore data*, and *Verify data*. The specialized tasks and outputs are presented in

Table 5.2. It is noted there are implicit activities when exploring data: hypotheses modeling and testing (descriptive analytics), followed by discovering data mining opportunities (predictive analytics). These exploratory activities are analogous with the concept investigation and design activities in NPD, and shall be differentiated from the later *Modeling* stage of the CRISP-DM. Therefore, we explicitly add these activities in the *Data Understanding* stage and term them as *Generate and test initial hypothesis*, *Investigate feasibility of predictive analytics*, and *Discover repeatable analytics services*.

Table 5.2 Data understanding stage in CRISP-DM (Chapman et al., 2000)

Generic Task	*Specialized Task (Process Instance)*	*Output*
Collect initial data	Acquire the data listed in the project resources. This initial collection includes data loading, if necessary for data understanding. If multiple data sources are needed, integrating data could be an additional issue here.	List the datasets acquired, together with their locations, the methods used to acquire them, and any problems encountered.
Describe data	Examine the "gross" or "surface" properties of the acquired data and report on the results.	Describe the data that has been acquired, including the format of the data, the quantity of data, the identities of the fields, and any other surface features which have been discovered. Evaluate whether the data acquired satisfies the relevant requirements.
Explore data	This task addresses data mining questions using querying, visualization, and reporting techniques. These include distribution of key attributes relationships between pairs or small numbers of attributes., results of simple aggregations, properties of significant sub-populations, and simple statistical analyses. These analyses may directly address the data mining goals; they may also contribute to or refine the data description and quality reports, and feed into the transformation and other data preparation steps for future analysis.	Report first findings or initial hypothesis and their impact on the remainder of the project. If appropriate, include graphs and plots to indicate data characteristics that suggest further examination of interesting data subsets.
Verify data quality	Examine the quality of the data, addressing questions such as: Is the data complete? Is it correct, or does it contain errors and, if there are errors, how common are they? Are there missing values in the data? If so, how are they represented, where do they occur, and how common are they?	List the results of the data quality verification; if quality problems exist, list possible solutions. Solutions to data quality problems generally depend heavily on both data and business knowledge.

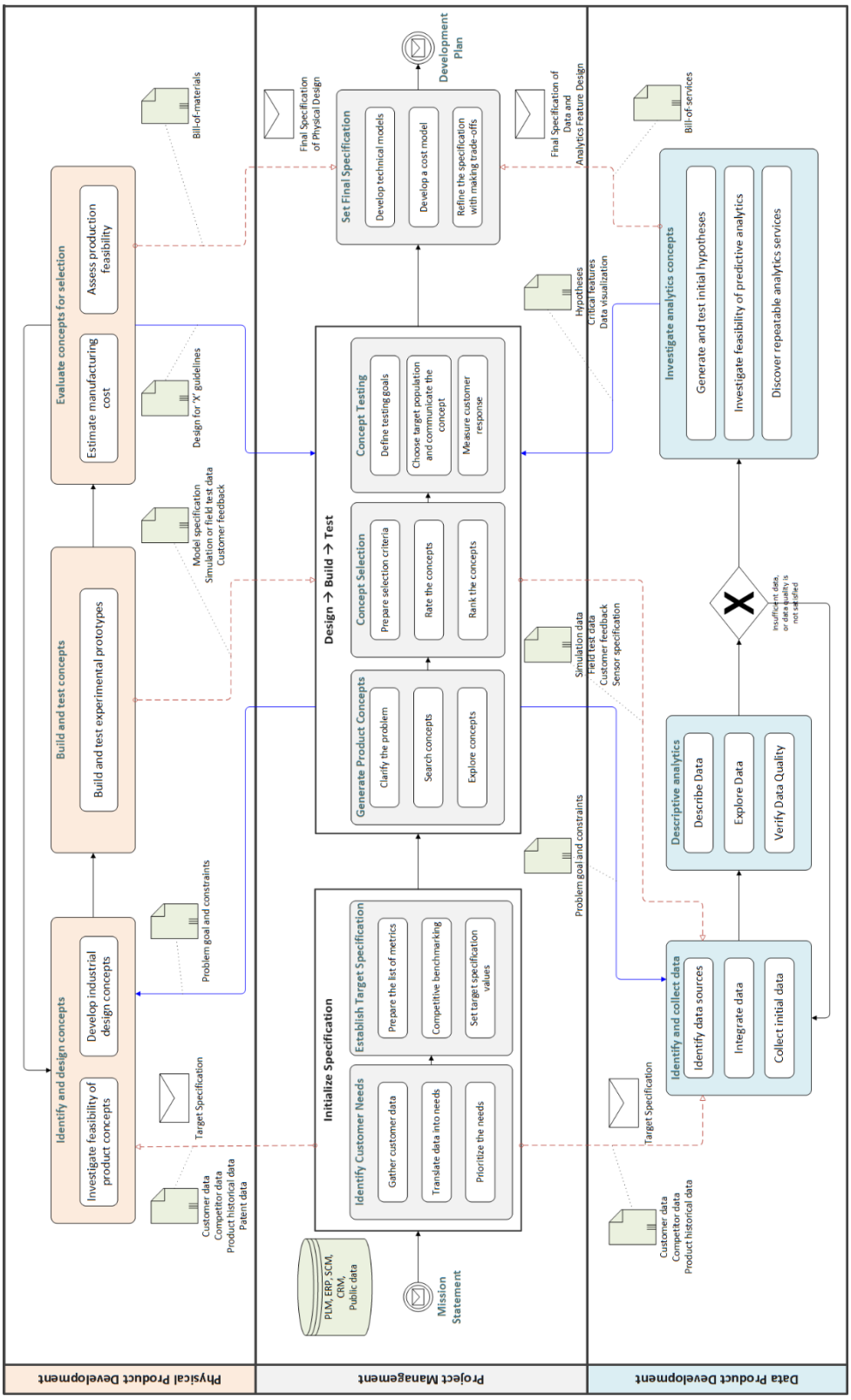


Figure 5.1 Process model for new product development with data-driven features (NPD^3) – Concept Development

In summary, as shown in Figure 5.1, the engineering activities are grouped as *Identify and design concepts*, *Build and test concepts*, and *Evaluate concepts for selection*; similarly, the data science activities are grouped as *Identify and collect data*, *Descriptive analytics*, *Verify data quality*, and *Investigate analytics concepts*. This setting structures the time lag for data scientists' activities compared to the engineers' activities.

Specifically, in Figure 5.1, the upper lane represents the engineering team, who focus on the engineering tasks for the development of the physical product. They translate the customer needs into the technical specification and use the technical specification to come up with the optimal solution for the physical design. Engineers usually employ well-established *Design for 'X'* principles (e.g. Design for Manufacturing and Assembly, Design for Environment, etc.) to evaluate and refine the product concept (Li and Roy, 2018). The final specification includes a bill-of-materials of the physical components and target values of their properties. The *Identify and design concepts* task has larger information integration workloads while the *Evaluate concepts for selection* has larger information dissemination workloads.

The lower lane represents the data science team, who focus on data processing and analytical modeling tasks for the development of the data product. They translate the customer needs into the data specification and use the data specification to come up with the optimal data analytics solution. Since data quality greatly impacts the analytics results, there must be a gateway to go/kill the decision to the *Investigate analytics concepts* task. The final specification includes both the data specification and the analytics feature specification. Similarly, a bill-of-services shall be included if the analytics feature can be further decomposed into reusable services. Intuitively, the *Identify and collect data* is dominated by information collection workloads and the *Investigate analytics concepts* task is dominated by information dissemination workloads.

The middle lane in Figure 5.1 represents the project management (PM) team, who follow a stage-gate-based NPD process. It takes a *Mission Statement* as input and produces the approved *Development Plan*. The first two tasks (*Identify customer needs* and *Establish target specification*) involve the marketing team, management team, customers, and other stakeholders. The engineers and data scientists participate in these preparation stages, and their collaboration is mainly brokered by the PM team. The dominance of outgoing flows indicates the information brokerage and dissemination role of these tasks. The design-build-test task group comprises of the core activities with which the engineers and data scientists work to collectively solve the problem. Detailed tasks are conducted in the individual team activities. Engineers and data scientists can use face-to-face communications if the organizational structure and geolocation allow, and the PM team can focus on ensuring the coordination of the tasks. This task group is also where the high rate of iteration takes place. The last task, *Set final specification*, again involves stakeholders from many disciplines to complete the development plan. The dominance of incoming flows indicates the information integration workloads of the task.

Note that the engineering and data science tasks are coordinated by the PM design-build-test tasks; hence, there are two implicit gateways (for project decomposition and integration) located before and after the design-build-test task group. It is also noted that test data from a simulation model, a physical prototype, or a field test can only be obtained after such a model/prototype has been built. Therefore, there is a message flow from the *Build and test concept*s task to the *Identify and collect data* task of a later iteration for data scientists. In addition, the sequence flows across the discipline boundaries also carry the necessary message information; we do not draw explicit message flow symbols for a clearer representation.

5.2 Case Study: *Smart UAS Development*

The initial NPD^3 model sets up the key tasks and main data/information flows, but we need to understand, in more detail, the content of these information flows and the team interaction patterns. In this section, we report on a project that utilized the NPD^3 approach to develop an unmanned aircraft system (UAS) that integrated advanced analytics within an unmanned aircraft vehicle (UAV) and its supporting systems, which we term a "Smart UAS."

The project was carried out during December 2016 to April 2017. The information for this case study consisted of weekly semi-structured observation notes. In addition, a product lifecycle management (PLM) system based on the sPLM concept was deployed for the team to centrally store the project artifacts (e.g., project weekly meeting minutes, 3D models, simulation data) and this project documentation was also leveraged to analyze the case study. The details of the PLM system development for UAS is reported in Chapter 6. Below we report the project requirements, team formation, and the data infrastructure to support the team collaboration. We then report on the concept design process, including discussion of the challenges faced by the project team, the concept testing that was performed, and how our NPD^3 approach was leveraged within the case study.

5.2.1 The UAS requirement, team formation, and project management

The smart UAS was designed for a *Water-Quality-Sampling* application that was requested by a civil engineering scientist. The usual practice in this area is to collect small water samples for lab analyses because many water properties cannot be measured in the field (Ore et al., 2015). If the properties can be measured in the field, they require an onsite monitoring system or a suitable vehicle to carry the instruments. In our case, the scientist requested a UAS to measure the water properties including *temperature*, *pH*, *dissolved oxygen*, etc. A UAS platform could access hazardous

environments, be more flexible than an onsite water monitoring system, and be faster than other vehicles (e.g. a boat). Most importantly, if properly designed, a UAS platform could be a cost-effective solution with the capability to adapt itself to conduct different missions. The overall requirements and the initial system specification are shown in Table 5.3.

Table 5.3 The UAS Requirement and the Target Specification

Requirement	*Target Specification*
Is lightweight to be carried and operated by a single scientist	The total weight is less than … kilograms with all the necessary payloads.
Is cost-effective	The total cost is less than … dollars.
Can hover over the water area	The UAV can hover 1 meter above the water surface without moistening the onboard payloads.
Is safe and has certain levels of autonomy	The UAV can detect and avoid a static or dynamic obstacle within a radius 5 meters surrounding it.
Can measure a range of water quality properties at predefined locations	The UAV can measure pH, temperature, dissolved oxygen using onboard sensors. The accuracy of measurement should fall between … and …
Can collaborate with other UAVs to achieve a mission	The UAV can work with at least one another UAV to simultaneously measure the water quality properties at a predefined location without sacrificing the safety.
Can quickly adapt to different water sampling missions	The UAV can replace with different sensors to sample different water quality properties.

Table 5.4 The Multidisciplinary Team

Role Group	*Role Description*	*Team members*
End user	Providing the domain knowledge regarding the water quality monitoring application.	One professor from civil engineering
Mechanical design and manufacturing	Designing the UAS mechanical parts and their configurations; 3D modelling and 3D printing of the custom-made parts.	Two professors from mechanical engineering One PhD student and one Master student from mechanical engineering
Electrical design and software development	Designing the UAS autopilot, communication and control, and autonomy algorithms.	One professor from computer science and one professor from control engineering
Data architecture and analytics	Designing and implementing the data management architecture and data processing platform; Conducting data processing and data analytics.	Two PhD student from mechanical engineering with the background of industrial engineering and data science
Pilot/Operator	Operating the UAV for test flights and mission flights.	One FAA certified UAS remote pilot
Project management	Overseeing and coordinating the project.	One principal researcher of the project
Project sponsor and industrial advisor	Providing advices and feedbacks from the industrial perspective	One industrial expert from sensor industry

There were twelve people working on this smart UAS project. As shown in Table 5.4, the team was comprised of researchers on the mechanical, electrical and data groups, as well as a remote pilot, a project manager, and an industry expert. The author of this dissertation was part of the team to help building the data infrastructure, providing data analytics guidance, and coordinating the project management. In the project kickoff meeting, we presented the NPD^3 diagram of Figure 5.1 to the project

team and explained the NPD model, the CRISP-DM model, and the integrated model. The NPD^3 model provided a common language and guidance to both engineers and data scientists, who otherwise are not familiar with the process used by their counterparts. We then documented the observations via weekly semi-structured notes throughout the remaining project time.

A data model was developed to capture the metadata of the generic elements and their relationships of the UAS architecture. The data model was derived based on the concept of the *Smart Component* data model discussed previously, this abstract model facilitated data storage, access, exchange, and tracing of all the data generated throughout the project, and the details of the model are reported in Chapter 6.

5.2.2 The UAS development

There were not many UAS-based water-sampling applications available when this project was started. In the project preparation stage (the first two weeks), a large number of articles in the fields of infrastructure management, environment monitoring, and traditional water-sampling methods were reviewed and studied. The recent research topics on UAS were also explored from technical publications. For example, publications on the International Conference on Unmanned Aircraft Systems (ICUAS) during 2013-2016 timeframe indicates topics such as UAS applications, navigation, path planning, control architectures, and simulation were constantly the top research areas. Other data sources included the product specifications from UAV and sensor vendors, the patent database for water-sampling mechanism design, and government data regarding water quality monitoring, etc.

The sharing of this information, including the system requirements, literature analyses, and other publicly available information, together with the initial target product specification were coordinated by the project management team and able to be accessed by both the engineering and data analytics groups for concept development. In the early concept exploration stage, the project team met frequently to

brainstorm the possible concepts, during which the domain knowledge had to be exchanged.

Each concept needed to consider a suitable configuration of the UAV hardware (air frame, avionics, payload, and power system), autonomy functions (state estimation, obstacle avoidance, etc.), and data communication methods. In order to leverage the potential of the data analytics, several questions were consistently asked by the team as they generated each new concept:

- *Is the current knowledge sufficient to capture the real-world dynamics (for example, the water area the UAV will fly over)?*
- *If not, can the problem in hand be solved by a data-driven modeling approach, and with what hypotheses?*
- *What data should be collected and how often should the data be collected?*
- *What sensors should be used and what parameters are required?*
- *How to decompose the decision-making process of an autonomy function?*
- *What are the repeatable/reusable analytics services can be adapted for future applications?*
- *Where to implement the analytics services, onboard or offboard? What are the physical constraints?*

These questions occurred across all levels of the concept development process. For example, it was difficult to preestablish a model for the target flight environment to deploy a UAV to a water area. We needed to check the terrain, water surface, weather dynamics, and any possible surrounding obstacles. The establishment of such an environmental model needed significant effort to work with the many external data service providers, for instance, the UTM (UAS Traffic Management) services. The algorithms to map the environment could be implemented either at the ground control station computer or onboard the UAV equipped with LiDAR sensors or vision cameras. Furthermore, these functions should work independently without affecting the water-sampling, the main function of the UAS. This implies the data infrastructure and communication protocols had to be co-developed with the UAS hardware and control software at the system architecture level. At the component level, a challenge the

team faced was '*What if the UAS is used in a GPS-denied environment where the GPS signal is no longer available*'. In this situation, two alternative concepts could be viable: (1) using other types of global positioning systems to provide the GPS-equivalent data; (2) using a completely different localization method, for example, a vision-based or a LiDAR-based system, to predict the desired state variables. In the first case, another positioning system (such as GLONASS) providing the same GPS format data solves the problem, the software and all the data-processing functions for state estimate are not necessarily changed. In the second case, additional sensors, processors, software, data-processing functions have to be built into the design, which means we need to review and revise the system architecture.

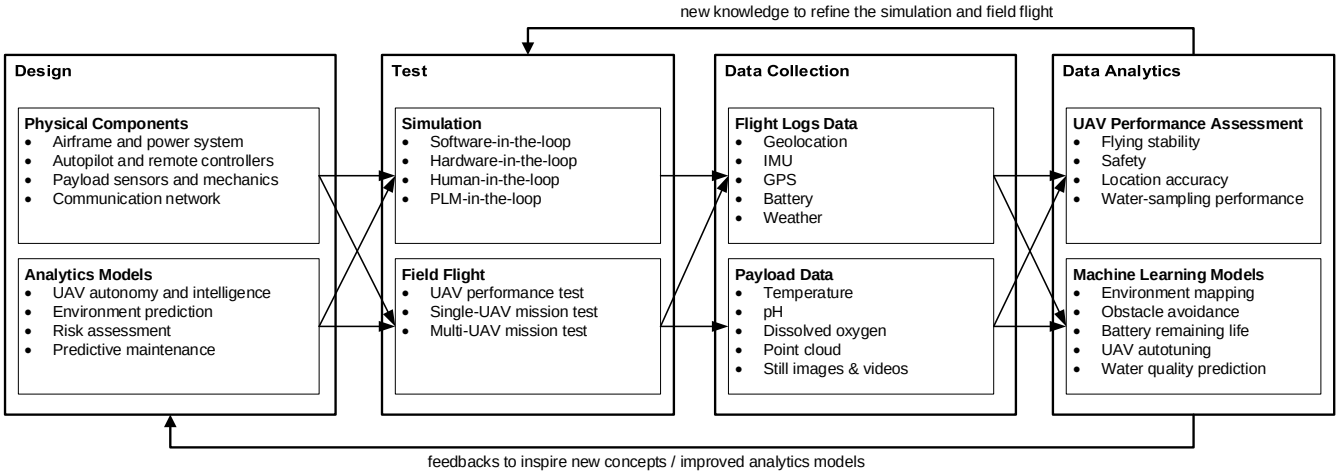


Figure 5.2 The UAS design, simulation, test, and data analysis

Similar to the Nest Thermostat development, the team set up a hybrid simulation environment that incorporated a UAV simulator, communication hardware, and the PLM system as a data repository to automatically store the test mission plans and process the mission data so as to provide performance analysis and train the machine learning models (Figure 5.2). The simulator used the same autopilot controller firmware as was used by the real UAV so that the simulation settings could be used for field test flights. The data scientist could work with the simulation data to explore the data and build initial

machine learning models before any real data had be collected from the field flights. Then, the validity of the models could be tested by the field tests. The data was used for either diagnosing the UAS performance (e.g. flying stability due to inappropriate tuning or signal interference) or for historical data to build predictive models (e.g. obstacle recognition and avoidance). The results of the data analytics were in two folds: (1) feedback to the next design iteration to inspire a new design, and (2) analytics models directly improving the current concept. This proves our initial observation of the two data-centered (*data-informed* and *data-driven*) product design paradigms from the Nest Thermostat smart product. The latter is an interesting "self-improvement" effect, in that it is a unique characteristic in a data-driven product. For example, we can recall the smart thermostat case, where more installations and usage generate more data that could be used improve the auto-schedule feature; similarly, more flight scenarios could enhance the UAV obstacle avoidance capability.

Table 5.5 Several UAS Concepts

Concept	*Concept Description*	*Characteristics*
Concept #1	Quadcopter with autopilot ADS-B for sense-and-avoid Water-quality sensors Cloud data storage	Low cost Collaborative sense-and-avoid capability No to low level autonomy
Concept #2	Quadcopter with autopilot and onboard computer LiDAR for detect-and-avoid Water-quality sensors Onboard and cloud data storage	Affordable Non-collaborative sense-and-avoid capability Low level autonomy
Concept #3	Octocopter with autopilot and onboard computer ADS-B and LiDAR for detect-and-avoid High-precision water-quality sensors Onboard and cloud data storage	Expensive Collaborative and non-collaborative sense-and-avoid capabilities Medium level autonomy High payload capacity

Several sample concepts are presented in Table 5.5. Both the engineers and data scientist had their domain-specific requirements. For example, a set of well-established *Design for 'X'* principles (e.g. design for manufacturing and assembly, design for sustainability) were used by the engineers to evaluate and refine the product concepts. The product bill-of-materials for physical components was critical to determine the selection of raw materials, manufacturing tools and processes, as well as the assembly/disassembly and recycling methods. For instance, we can 3D print the custom-designed

compartments for battery and water-sampling sensors. Similarly, data scientists employed a set of measurements, including data quality, prediction accuracy, computational costs, the capability to incrementally update with new data, to screen the analytics models. A bill-of-data and bill-of-services for data analytics models were also critical to determine which data analytics techniques should be employed in the downstream processes. Collectively, the criteria for concept ranking and selection took the functionality, level of autonomy, cost, degree of modularity, and regulation requirements into consideration. Here, the level of autonomy is an important criterion even though it may compromise the overall cost and the product modularity (because of redundant components and computation). The team chose the second concept that was overall affordable and was satisfied with the project requirement. This concept included a DJI F450 quadcopter air frame, an open-source PixHawk mini autopilot controller, and a LiDAR sensor for scanning the surrounding environment (Figure 5.3). A portable computer, Raspberry PI, was used as the companion to the autopilot controller in order to deploy a custom-designed artificial potential field (APF) algorithm (Yin et al., 2017). This algorithm dynamically generates obstacle-free paths based on the real-time LiDAR data.

No.	Description	Quantity
1	GPS Module with stand	1
2	Propellers	4
3	Lidar	1
4	DJI Brushless Motors	4
5	Bolts	4
6	Support for Lidar	1
7	Electronic Speed Controller	4
8	Bolts	16
9	Pixhawk Flight Controller	1
10	Power Regulator	1
11	Radio Module	2
12	Buzzer	1
13	Subassembly Base-Box	1
14	DJI Frame Flame Wheel F450	1

No.	Description	Quantity
1	Sensor Circuits	3
2	Raspberry Pi	1
3	Battery	1
4	Screws to fix Raspberry Pi	3
5	Bolts to fix the base-box on to the airframe	1
6	Lock Plate	1

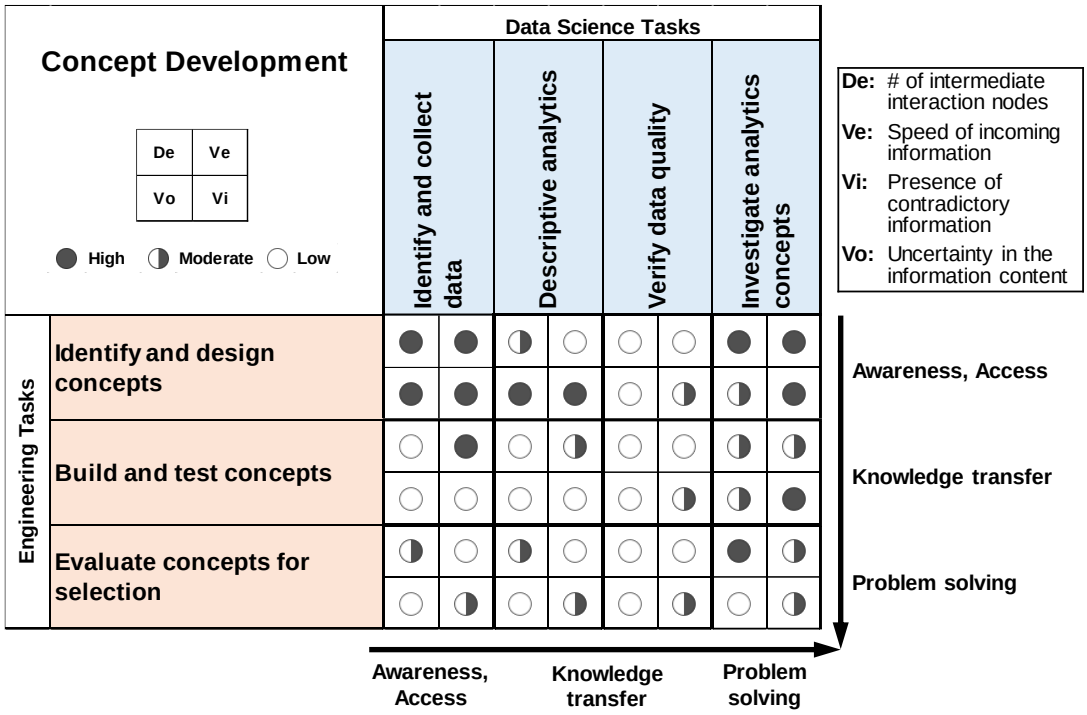
Figure 5.3 A UAS concept: the UAV components and 3D model

The final UAS specification included the hardware specification, software specification, data specification, and analytics model specification. System design and other later stages defined in the NPD and CRISP-DM were then be followed.

5.3 Observations of the Team Interaction Patterns and Characteristics

The UAS development project lasted for five months. The team interaction patterns were summarized and shown as in Table 5.6. Specifically, the rows in the table represent the phases of the project from a data science perspective and columns represent the phases of the project from an engineering perspective, and each cell represents a possible set of interactions.

Table 5.6 Interaction patterns and characteristics between engineers and data scientists



In analyzing the goals of the interactions, we followed Distanont et al. (2012), who noted that, in a collaborative product development network, one can view the interaction flows as one of four goals for

the interaction – *Awareness*, *Access*, *Knowledge-transfer*, and *Problem-solving*. Table 4 shows that as one moves further along the concept development process, the goal moves towards problem-solving. In the smart UAS project, a significant amount of interactions was needed to identify the data sources and interests during the early project phase. The multidisciplinary team was finally able to collectively deliver the water-sampling UAS platform with appropriate composition of physical components, compatible control software, and suitable data analytics pipeline.

Furthermore, in analyzing the characteristics of each information flow that occurred within our case study, we leveraged four attributes proposed by Krovi et al. (2003). The *Density (De)* is defined by the number of intermediate interaction nodes. The *Velocity (Ve)* refers to the speed of incoming information at an interaction node. The *Viscosity (Vi)* reflects the degree of conflict due to presence of contradictory information components at the interaction node. The *Volatility (Vo)* denotes the associated uncertainty in the information. At a high level, when the UAS project had to integrate and evaluate the various concepts, the presence of contradictory information increased because there had to be a compromise for the multiple performance measures. It was also observed that the speed of incoming information was initially high, then decreased but then increased later, once the simulation model started to generate data based on various trial settings. As the process progressed, more data and information was available and the design problem was more constrained; therefore, the problem became less uncertain.

With this framework, we can categorize the interaction for the flow of a specific phase combination. Below we describe the interaction patterns among the different combinations of engineering and data science project phases. As a starting point, we define each attribute to have three levels: *Low*, *Moderate*, and *High*:

1) *Identify and design concepts – Identify and collect data*: At the start of the project, a significant amount of interaction between engineers and data scientists was needed to identify the data

sources and interests. The data sources included market surveys, technical publications, patent database, government data, and manufacturer/vendor whitepapers. The velocity of incoming data was fast, the data/information present had a high conflict, and uncertainty was also high. In short, all four parameters were at the high level.

2) *Identify and design concepts – Descriptive analytics*: At this stage, the data had been collected, and the data science team was focusing on the data analysis. Hence, the interaction density was moderate and velocity was low. However, the viscosity and the volatility were high because the two groups had a different understanding with respect to the large amount of data from the different sources. For example, it was difficult for the data scientists to understand the meaning of each column presented in the flight log data.
3) *Identify and design concepts – Investigate analytics concepts*: At this stage, the data science team started to generate analytics concepts which in turn affected the development of the physical concepts. The interaction density again became high, the velocity and viscosity were also high since more data and information had become available. For example, the localization function required data from different sensors for GPS-friendly environment versus the GPS-denied environment. The concept designs for the sensor systems and the analytics models were mutually affected. The volatility was kept low to moderate.
4) *Build and test concepts – Identify and collect data, Descriptive analytics*: At this stage, the data sources were mainly from the simulation, field test, and the customer feedback. The data formats had been determined and the data stream processing could be automated to some extent. With both the physical and analytical concepts being built into the prototypes, descriptive analytics were conducted on various testing scenarios. Sensitivity analysis was conducted to identify the impacting variables on the product performance. The interaction density was therefore low, the velocity was high to moderate. There were low viscosity and volatility.
5) *Build and test concepts – Investigate analytics concepts*: At this stage, the data science team refined the previous analytical concepts, to generate and test new analytical concepts for the next iteration. The interaction density and velocity were moderate to high, the viscosity was high but the volatility kept low to moderate.
6) *Evaluate concepts for selection – Identify and collect data, Descriptive analytics*: At these stages, the product concepts had been filtered to a limited set, and the focus of data scientists had turned to a new iteration to collect data for product performance analysis – in order to provide guidance

in building a closed-loop product operation for continuous improvement. The interaction density and viscosity were moderate, the velocity and volatility were low.

7) *Evaluate concepts for selection – Investigate analytics concepts*: At this last stage, both groups determined the final concepts. The interaction density and viscosity increased again because of the integrated evaluation and there had to be a compromise across the multiple performance measures. For example, collecting more data was beneficial to the analytics model development, however, this implied the sensors and the controllers needed to work in higher frequencies. Thus, it had a negative impact on the battery power. The interaction velocity was moderate. The volatility was low because most uncertainties had been eliminated and there had been risk mitigation plan in place.
8) *All engineering tasks – Verify data quality*: At these stages, the data had been cleaned, processed, and descriptive analysis results are generated and presented. Data scientists needed engineering experts or other stakeholders to verify the data quality to prepare the datasets for the following analytical concept development tasks. The interaction density, velocity, and volatility were low. Since this was always a go/no-go decision-making point, the viscosity was moderate to high.

5.4 Decomposition of Information Content for Individual/Subgroup Tasks

In this section, we discuss, from a theoretical perspective, the details of the information flows related to an individual task or group of tasks. To show these input and output flows, we employ an IDEF0-based notation which decomposes the information related to unit collaborative design activity into four categories: *Intra-disciplinary design information (I)*, *Cross-disciplinary design information (C)*, *External design information (E)*, and *Design information output (O)*. This notation was originally proposed by Austin et al. (1999) termed as IDEF0v, in order to facilitate a collaborative building-design process, while the IDEF0 (Integrated computer aided manufacturing DEFinition for function modeling) technique was developed in order to better communicate and analyze manufacturing systems in an attempt to improve productivity.

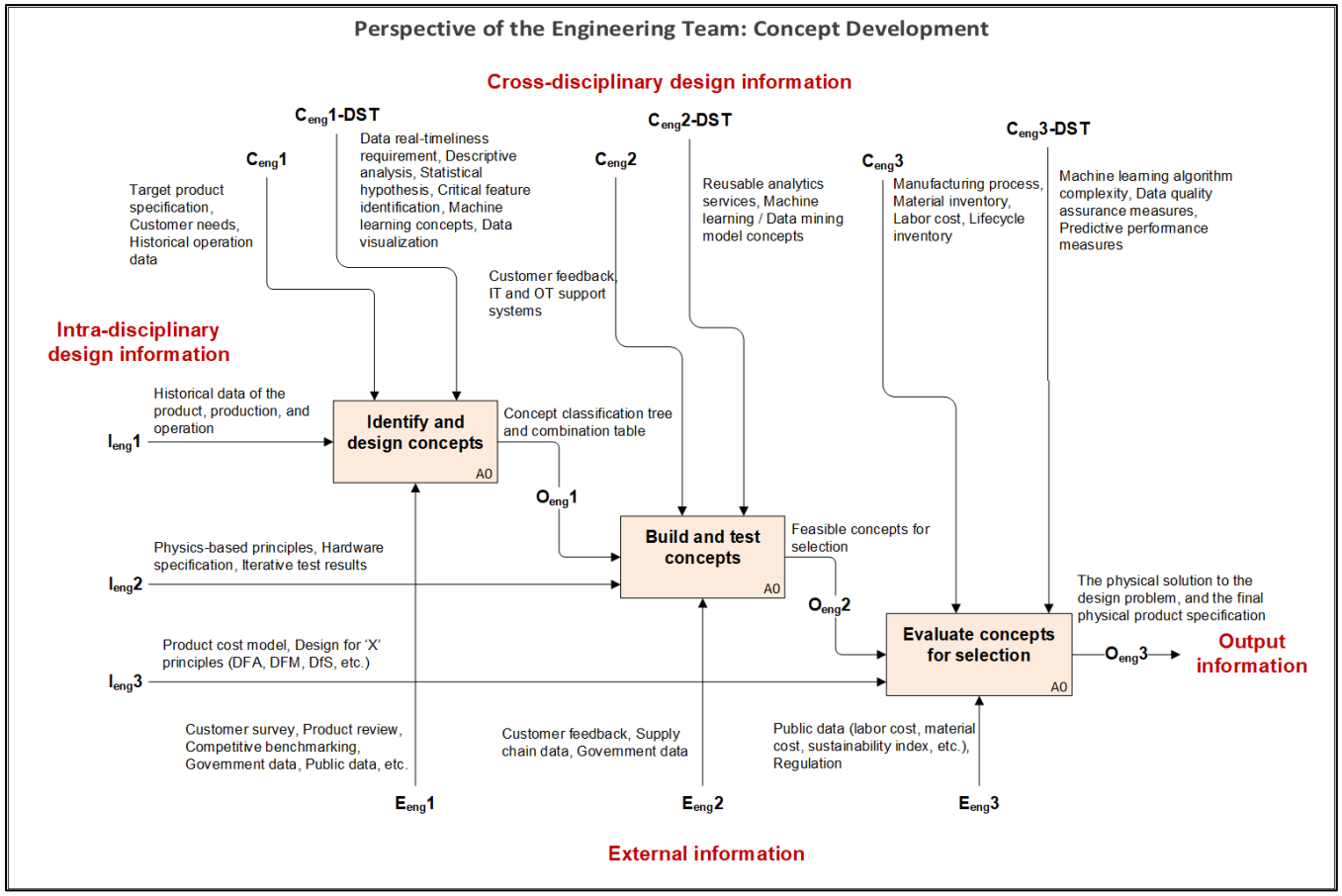


Figure 5.4 Information flow from the engineers' perspective

The information content of each information flow is identified by revisiting the standard activities defined in the NPD and the CRISP-DM, as well as the activities we found during our smart UAS project. For instance, the intra-disciplinary input information for the *Identify and collect data* stage consists of historical data of the product and production, while the output information includes the concept classification tree and combination table. By this way, the information flows for the engineering and data science activities are elaborated in Figure 5.4 and Figure 5.5. The intra-disciplinary, cross-disciplinary, external, and output information for the engineers are encoded as **I_{eng}1~3**, **C_{eng}1~3**, **E_{eng}1~3**, and **O_{eng}1~3**. We explicitly encode the potential cross-disciplinary information received from the data scientists as **C_{eng}1~3-DST**. On the data scientists' side, **I_{dst}1~4**, **C_{dst}1~4**, **E_{dst}1~4**, **O_{dst}1~4**, and

C_{dst}1~4-ENG are the intra-disciplinary, cross-disciplinary, external, output information, and cross-disciplinary information explicitly from the engineers.

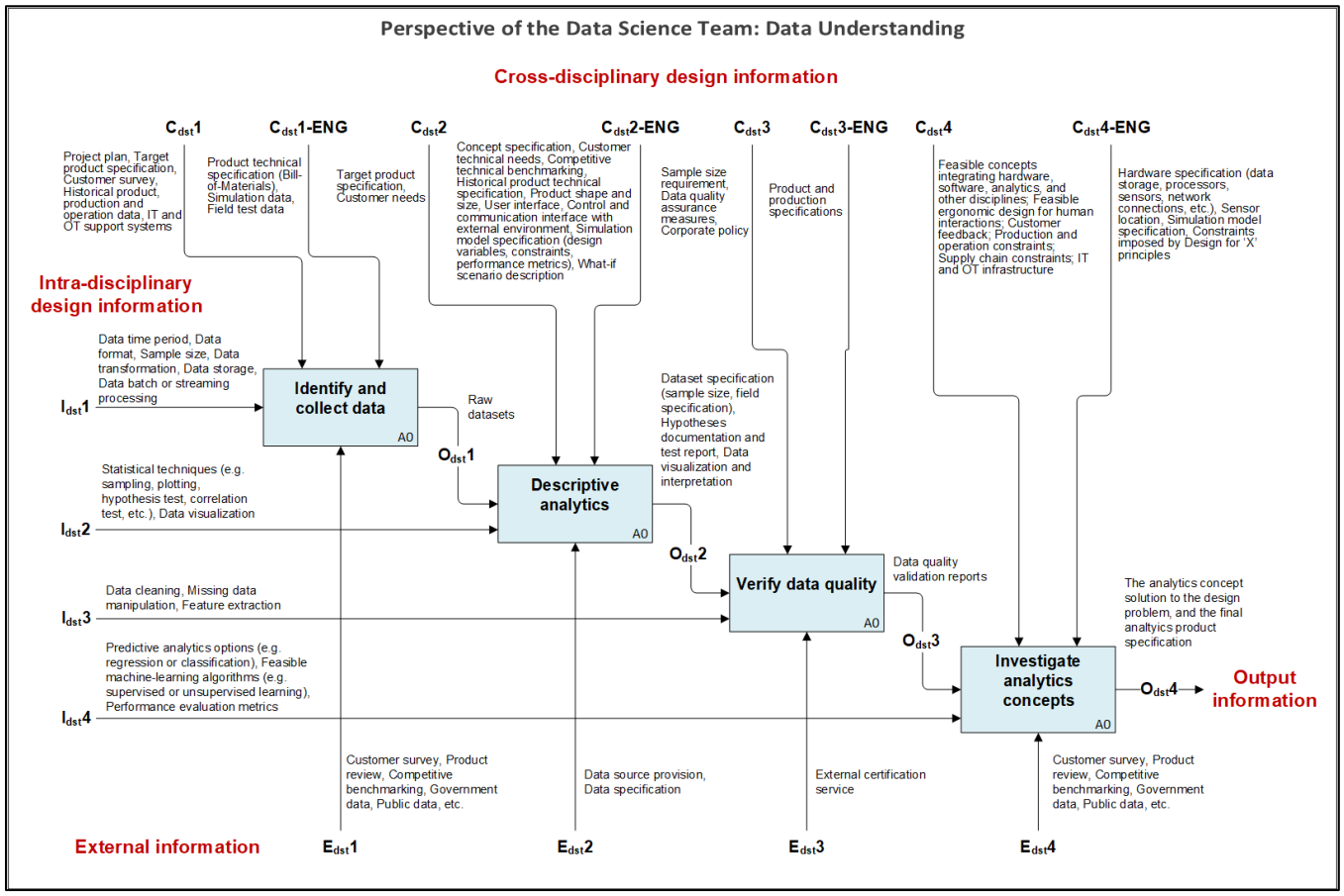


Figure 5.5 Information flow from the data scientists' perspective

This information decomposition reveals several interesting factors of the information dependency between the engineering and data science groups. First, the external and cross-disciplinary information shows the shared information between the two groups (for example, product specification, customer feedback, and publicly available data), indicating that a common dedicated team, or a higher-level project management team (if there is one), could help broker this information. In the smart UAS project, the project manager, the industrial advisor, and the end user indeed helped to coordinate tasks for the collection and dissemination of these shared data. The PLM implementation of the high-level NPD[3]

model also helped the data/information sharing. Second, the cross-disciplinary information coming from the two groups indicates engineers and data scientists may need to directly communicate with each other for effectiveness and efficiency, suggesting that an appropriate organizational structure or geolocation arrangement between the two groups would be helpful. In our case study, the engineers, the data scientists, and the remote pilot were from different departments, which created some time schedule and communication challenges. For example, there were situations where the data scientists were waiting for new test data but the pilot was not available for field testing the new engineering design.

Furthermore, the information content in the cross-disciplinary flow not only needs to be aware of and accessed by each group, but also transfers domain-specific knowledge to the counterpart for collective problem-solving. The output information of each activity is not only for the next activity within the same discipline, but might also be consumed by the collaborative tasks in the other discipline. This supports what Cooper (2014) had argued: an effective process requires each subsequent task to maximize the utility of the stable information available from the previous task. However, it was observed that the analytical concept generation was always at least one step behind the physical concept generation unless the data could be obtained from an existing data source. This implies a dependency between these two concept generation processes. Hence, simulation with appropriate assumptions becomes a critical method to synchronize these two processes. This is consistent with previous finding regarding the development of a data-driven manufacturing system (Li and Roy, 2015). Finally, Figure 5.1, Figure 5.4, and Figure 5.5 together provide a more complete view of the NPD^3 model.

5.5 Summary

This chapter proposes NPD^3, an integrated process model for new product development with data-driven features. We revisit the classic NPD process model and a well-adopted data analytics process

model, CRISP-DM, to understand the key tasks prescribed for the engineers in a physical product development team and the data scientists in a data products development team, respectively. The NPD^3 model was then evaluated within a case study of the creation of a smart unmanned aircraft system. The results of our case study demonstrate that there was cross-disciplinary design information required by the engineers as well as the data scientists, and that it was critical that direct interactions and messages were exchanged between the two groups, with a project management group acting as a mediator to guide the collaboration across the team. In addition, the project management group could help ensure that the needed external design information be shared between two disciplinary groups.

The timing and contents of information exchanged between the two groups facilitate the information awareness and access, knowledge transfer, and problem-solving. We use four attributes – the number of interactions, the speed of information, the contradictory information, and the uncertainty in the information – to characterize the interaction patterns. At the beginning of our project, a significant number of interactions were needed to identify the data sources and interests. As the process progressed, more data and information was available and the design problem was more constrained; therefore, the problem became less uncertain. When it came to the integration and evaluation of the concepts, the contradictory information increased because there had to be a compromise for the multiple competing performance measures. It was also noted that the speed of information decreased at first but then increased once the simulation model started to generate data based on various trial settings.

Our integrated process model is encoded with BPMN notation so that it can be implemented for automation, in a business management system, e.g. a PLM system. The NPD^3 model and the PLM implementation provided the UAS development team a collaborative environment and data repository to facilitate effective data/information exchange, visual communication, and traceable decision-making. The integrated process model also provided a common language and guidance to both engineers and

data scientists, who otherwise are not familiar with the process used by their counterparts.

Chapter 6

Application: *sPLM for Unmanned Aircraft Systems*

6.1 Motivation

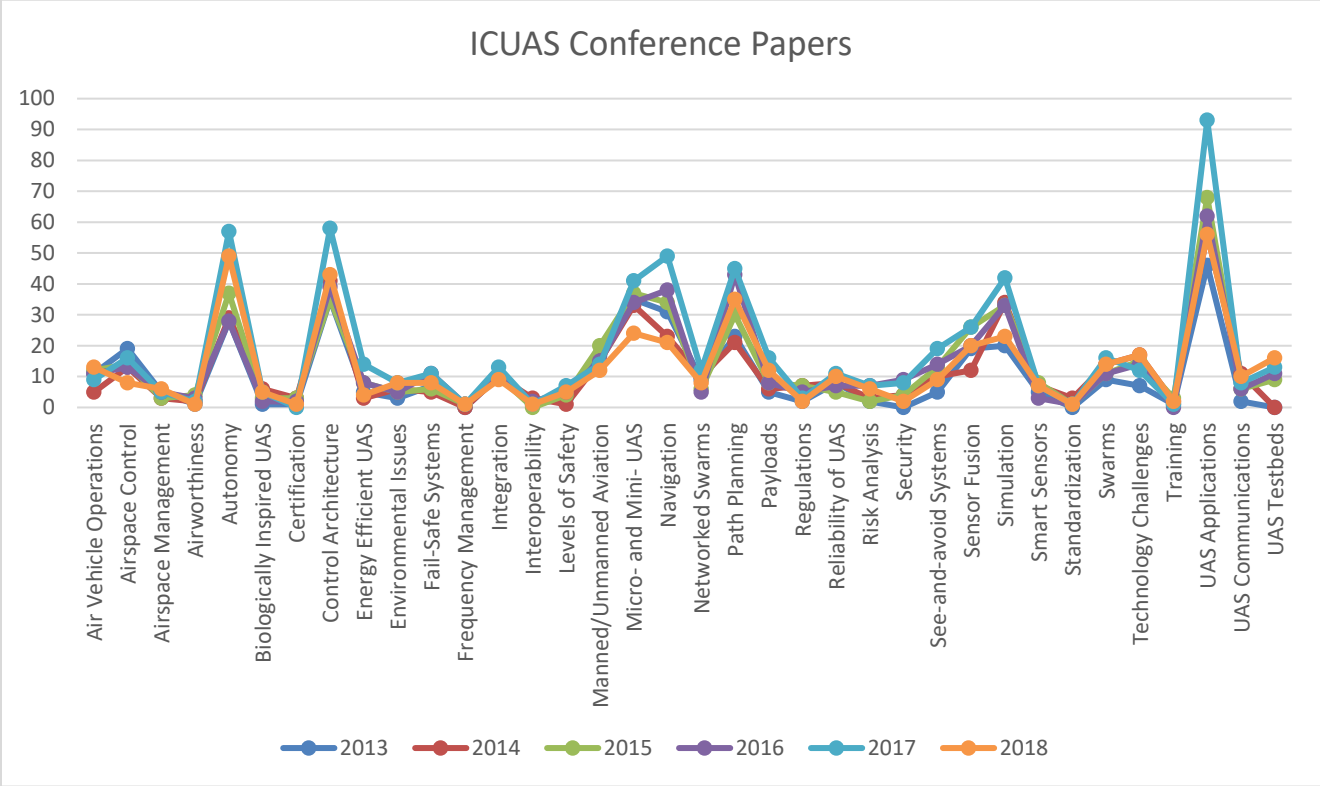


Figure 6.1 Hot research topics in UAS community (ICUAS, 2013-2018, Appendix C)

Unmanned aircraft systems (UAS) are becoming popular in civilian applications and have covered a broad range of areas including aerial photography/filming, infrastructure inspection, precision agriculture, environment monitoring and protection, disaster/crisis search and rescue, to name a few (Gupta et al., 2013). A UAS can collect accurate images of the planted crops that allow farmers to identify areas where crops need more attention to increase yields. A UAS platform can be four times faster than rope access or elevated platforms in inspecting wind turbines. For delivery companies, the last-mile UAS delivery operation could save tens of millions of dollars per year.

Figure 6.1 shows the research trend regarding the UAS topics in recent years. This data is based on the papers published at the International Conference on Unmanned Aircraft Systems (ICUAS) during the years 2013-2018. It has shown autonomy-related issues have gained increasing research interest, and the small UAS is being applied in many civil applications and scientific researches. More specifically, the top five hot UAS research topics include: *UAS applications* (381 papers), *Control architectures* (251 papers), *Autonomy* (228 papers), *Micro- and Mini-UAS* (204 papers), and *Navigation* (196 paper).

Every industrial application is different; however, all these applications share similar standardized workflows including UAS procurement, operations, mission planning, data acquisition and processing, data visualization and presentation (Figure 6.2). The ultimate goal is to seamlessly embed the UAS operations into day-to-day enterprise processes. It is important to design a modular UAS architecture so that the UAS can be easily reconfigured for various mission applications, minimizing the development time and cost while maximizing the mission performance.

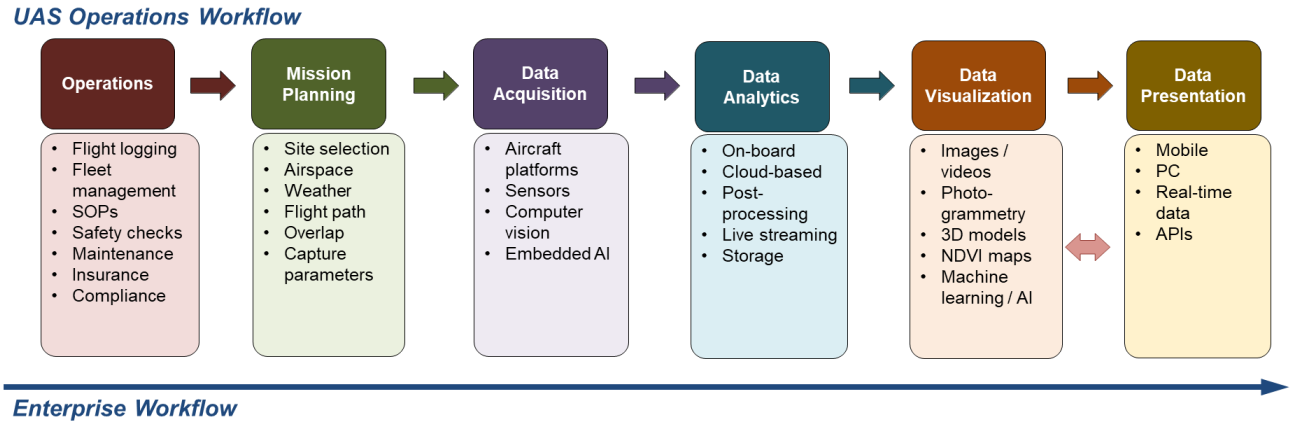


Figure 6.2 Standardized workflow for UAS operations (adapted from Colin, 2017)

6.2 PLM Needs and Challenges for UAS

As a complex system, a UAS as a whole and each individual component have their lifecycles. For any application, the development and operations of the UAS platform usually starts from a mission

requirement and goes towards its final operations and retirement. For example, a UAS-based water sampling system (Ore et al., 2015) has to consider the development of (1) the mechanism to capture water samples; (2) the sensors and algorithms for altitude approximation over water; and (3) software components that integrate and analyze sensor data, control the vehicle, drive the sampling mechanism, and manage risk (see Figure 6.3).

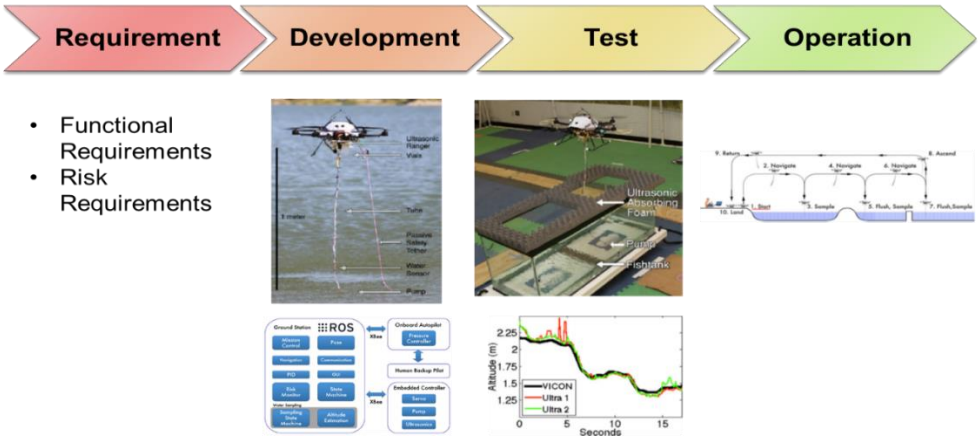


Figure 6.3 Lifecycle stages of a UAS application (adapted from Ore et al., 2015)

While safe UAS operations require transaction-oriented systems (e.g. UAS traffic management) for mission planning, execution, monitoring and controlling, a product-oriented information system (e.g. PLM) is equally important to trace the history of each component of the UAS along its lifecycle stages. A lifecycle management platform for UAS needs to accommodate all the data and activities that are involved in the system development and operations. PLM systems have been successfully used in developing and operating complex products/systems, but the best practice has not been fully established for the UAS industry.

This is because, it is costly yet often required for individual users or small user groups to tailor or configure an UAS as per their specific needs. Compared to the modularity of the physical components of a UAS, the modularity of autonomy is harder to achieve because the autonomy implementation is tightly coupled with other components and the data generated from the system. Furthermore, there are many

individual applications, algorithms, point-solutions that are needed be integrated. Besides, due to the small size of a UAS and its possibility to fly anywhere in any spatial direction, the mission planning and coordination are much more complex compared to other vehicle operations.

Recent literatures have suggested to extend the PLM role to the management of autonomous capabilities of vehicles (Bernabei et al., 2014). This approach suggests to extend the application of *Modelling and Simulation* from the initial phase of the vehicle' lifecycle to its utilization phase. Through this way, the autonomous capabilities are based both on previously simulated scenarios and on real-time computation, in order to adapt the behaviors of the vehicle to the real operational scenario. However, this model-based approach requires a model repository to store, retrieve, and update knowledge models during different phases of the vehicle's lifecycle. Below we discuss how the sPLM concept presented previously can be adapted for the UAS lifecycle management.

6.3 The sPLM Framework for UAS

A UAS, particularly, the unmanned aircraft vehicle (UAV) can be seen as a flying smart system. Since a smart product can be decomposed into two essential parts: the *Physical Components* to form the product's physical body, and the *Analytics Models* to implement the product's autonomy, we first decompose the UAS into these two dimensions. For UAS, the physical components can be an air vehicle, the propellers, the payloads, or the batteries; the analytics models can be an obstacle avoidance algorithm, an auto-tuning function, or a battery life prediction model. In order to achieve a particular mission, a UAS platform needs to be configured by appropriate UAV, payloads and accessories, and the compatible predefined algorithms and/or machine learning capabilities. The time domain for a UAS can be then focused on the operations stage only, i.e. pre-flight, in-flight, and post-flight, or focused on the entire lifecycle that includes design, manufacturing, use, and maintenance stages.

6.3.1 UAS Hardware, Autonomy, and Data

A UAS consists of five distinct elements (NATO, 2012): (1) the *Unmanned Air Vehicle (UAV)* element includes the air frame, power system, and the avionics required for flight control; (2) the *Payload* element includes the sensor systems, associated recording devices, and associated control/feedback mechanisms; (3) the *UAV Control System (UCS)* element incorporates ground and air control systems for generating, loading, and executing the mission and to disseminate information to various command, control, communication, and intelligence (C4I) systems; (4) the *Launch and Recovery* element incorporates the functionality required to safely launch and land the UAV; and finally (5) the *Data Link* element, which enables ground-air communication or air-air communication.

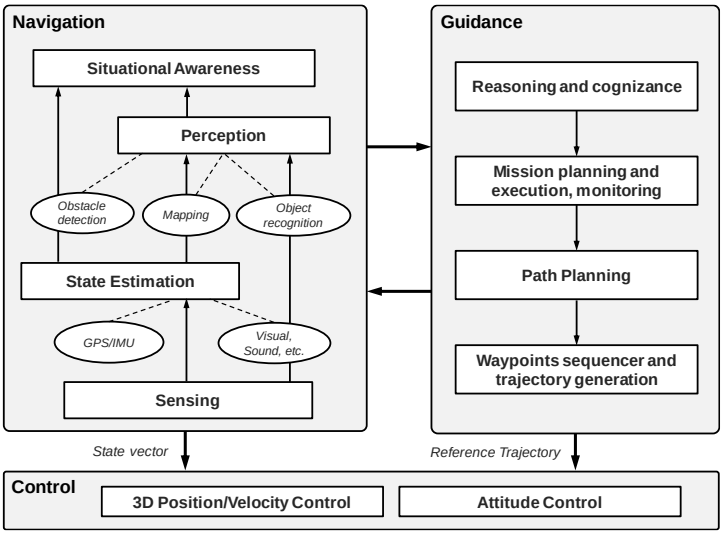


Figure 6.4 Navigation, Guidance, and Control systems for UAS autonomy (adapted from Kendoul, 2012)

The data-driven nature of a smart UAS arises from its transition from an automated system to an autonomous system. The autonomy of an UAS is defined as the UAS's own abilities of sensing, perceiving, analyzing, communicating, planning, decision-making, and acting/executing, to achieve its goals as assigned by its human operator(s) through a designed human-robot interface or by another system that the UAS communicates with (Huang, 2008). The autonomy enabling functions for a UAS

can be grouped into three subsystems: *Navigation*, *Guidance*, and *Control* (Kendoul, 2012). Navigation is the process of monitoring and controlling the movement of an air vehicle from one place to another. It is a highly data-intensive process involving data acquisition, analysis, extraction and inference of information about the vehicle's state and its surrounding environment with the objective of accomplishing the assigned mission successfully and safely. Guidance is the driver of the UAS that exercises the planning and decision-making functions to achieve the assigned mission or goal. It takes inputs from the navigation system and generates reference trajectories and commands for the flight control system. Finally, control is the process of manipulating the inputs to a dynamical system to obtain a desired effect on its outputs without a human in the control loop. The dependencies of these three autonomy components are shown as in Figure 6.4.

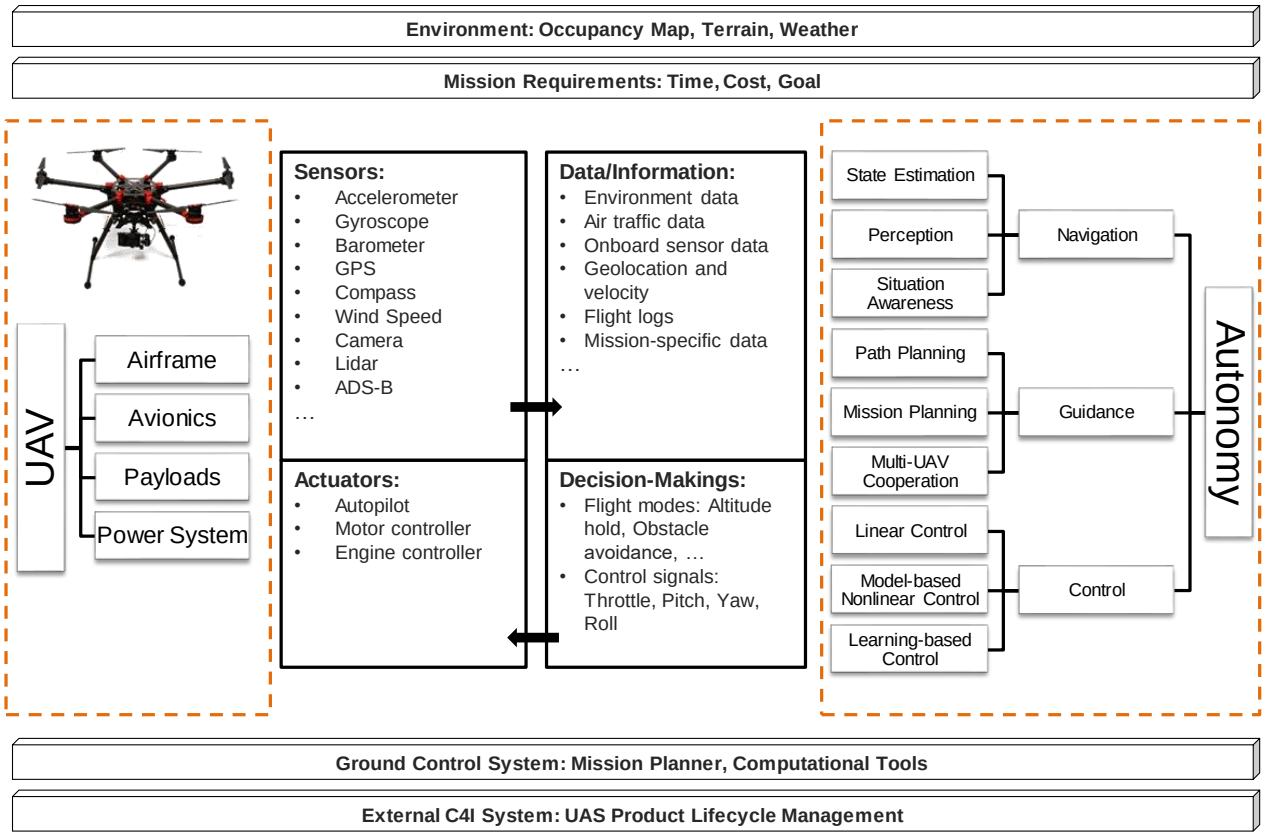


Figure 6.5 A generic architecture of an autonomous UAS (Li et al., 2017a)

The UAS data comes in many different sources and formats including flight logs, performance data, geo images, video files, LiDAR surveys, etc. As Kovar and Bollo (2018) suggested, the data is flown across: (1) *Physical Assets* that include the physical devices, the UAV, the batteries, the sensors, the remote controller, the ground control station, and on any computers used to maintain the UAV or process its data; (2) *Operational Procedures* that an operator prepares for a flight, conducts it, and manages the data after the flight. The phases for a mission lifecycle include: *mission planning*, *approval*, *execution*, *analysis*, and *delivery*. Each phase involves documentation, communication, or activities that can be collected and analyzed; and (3) *Communication Channels* that include the communication between the UAV, the environment, its supporting systems, and systems on the cloud.

Figure 6.5 depicts a smart UAS's generic architecture, which consists of its physical architecture, autonomy architecture, cyber-physical interfaces, and the supporting subsystems.

6.3.2 sPLM Architecture for UAS

With this decomposition, the sPLM framework established in previous chapters can be adapted to manage the lifecycle data and activities of a UAS. As shown in Figure 6.6, its core is the shared lifecycle management functions provided as web services. By unifying the data models for physical products and analytics models, these shared functions can be applied to all digital models of UAS devices, software, autonomy functions, and missions. The individual models can be versioned, tracked, and be composed with other compatible models, if needed. The rule and scoring engines embedded in the sPLM allow building and executing configuration rules, regulatory rules, and various machine learning models. We then abstract the general UAS development and operations processes and implement a set of template modules for UAS users including designers, manufacturers, operators, and other relevant stakeholders. These modules include:

- *User Management* for user authentication and qualification tracking (includes user profile, user

groups and roles, user permissions).

- *Mission Management* for mission planning, monitoring, fleet operation scheduling, and mission tracking (includes mission profile completion status, dashboard, environment data, mission planning and waypoints optimization, UAS fleet operations and monitoring).
- *UAV and Payloads Management* for managing device assets data and maintenance history (includes asset data of air vehicle, air frame, propellers, sensors, power systems, their engineering drawings and change histories, 3D models, and other relevant documents).
- *Autonomy Model Management* for software and analytics algorithm assets data management (includes algorithms and data analytics models for path planning, obstacle avoidance, etc.)
- *Datasets Management* for flight logs and simulation data analyses and visualization (examples: flight logs data, simulation data, any application data such as images or water samples captured by the UAS).
- *API/Connectors* for integrating supporting tools (examples: mission planners, CAD tools, data analytics tools, and external services).
- *Regulatory Documents and Rules Management* for regulatory compliance.

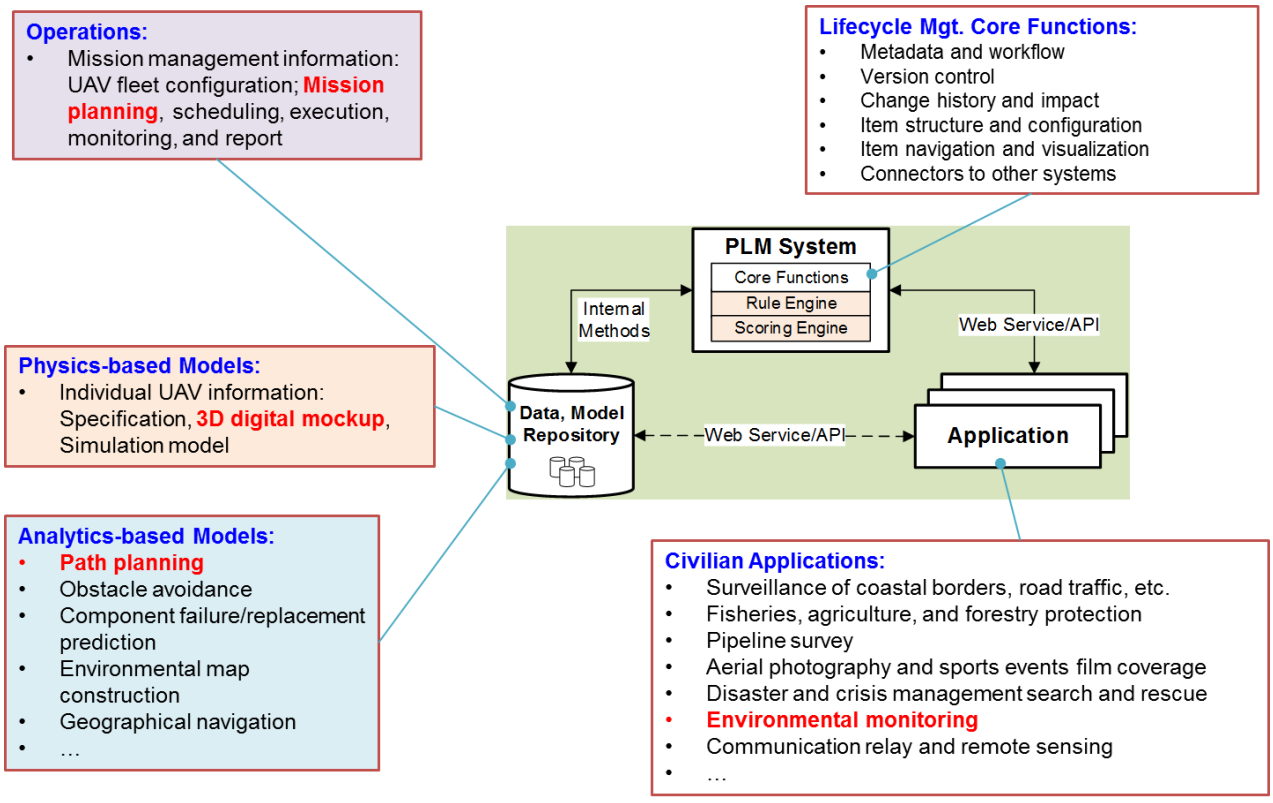


Figure 6.6 sPLM adaptation for UAS lifecycle management

6.3.3 UAS Data Model

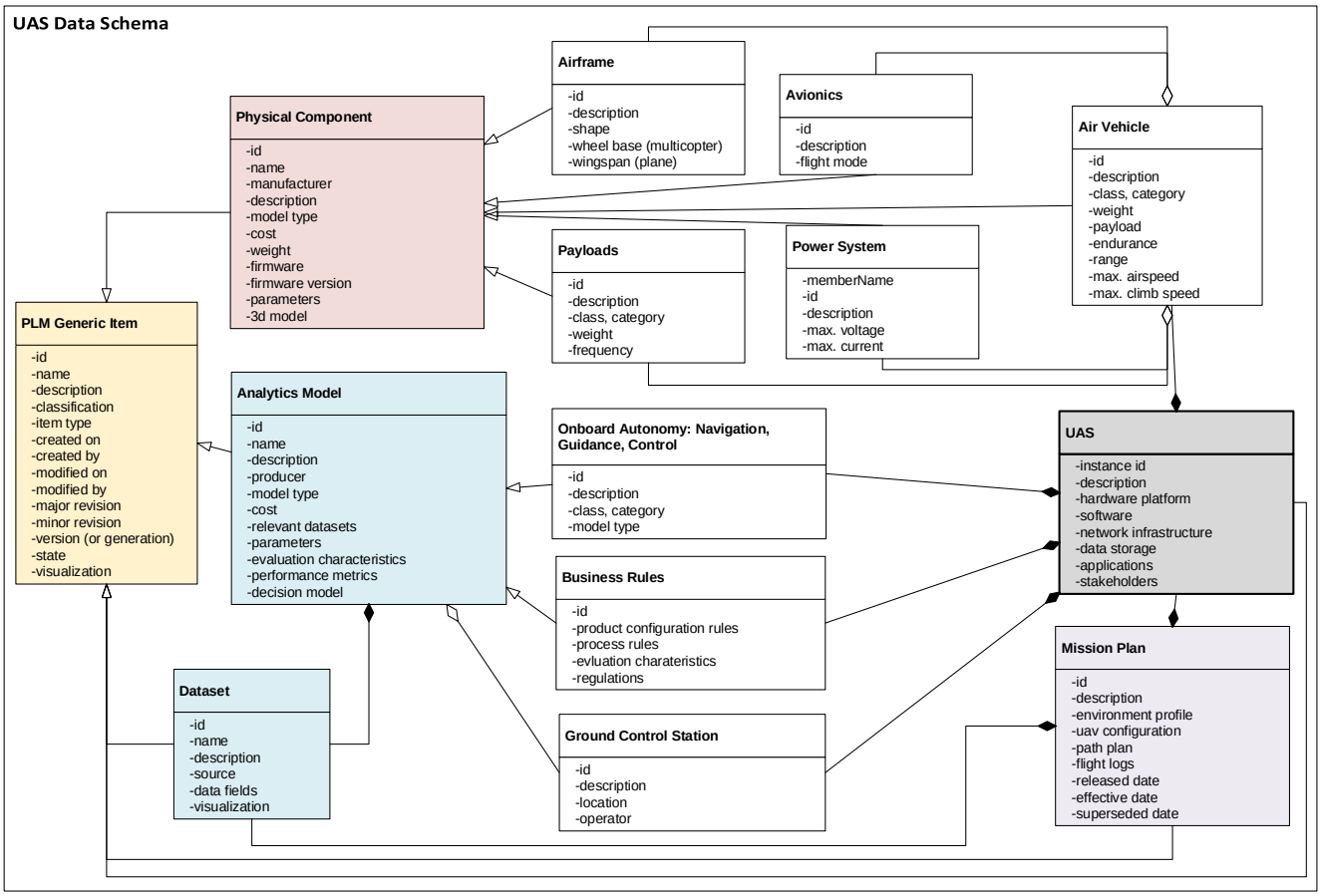


Figure 6.7 The UAS data schema (implemented in sPLM)

A data schema model (Figure 6.7) is developed to capture the metadata of the generic elements and their relationships as described in the UAS architecture shown previously (Figure 6.5). This abstract model is to facilitate data storage, access, exchange, and tracing all the data generated and flown throughout a UAS mission. The core classes of the UAS data model are described as follows:

- *PLM Generic Item*: This item is the root class of the sPLM system; all other classes inherit from this class and the children of this class.
- *Physical Component*: This item represents the physical components of a product to form its body. The classes for the overall air vehicle, the airframe and propellers, the avionics, the payloads, and the power systems are inherited from this class.

- *Analytics Model*: This item represents the analytical components of a product to implement its intelligence. All the autonomy-related functions (navigation, guidance, and control) can be implemented in different derivations of this class. Business rules, a specific case of analytics models, also inherits from this class and implements the regulatory rules.
- *Dataset*: This item represents the datasets that have been extracted, aggregated, cleaned, and structured from various sources of raw data. It can be a training dataset or a test dataset that provides the contexts to the analytics models built upon it.
- *UAS*: This item represents an application-oriented UAS that is composed of certain physical components and analytical components, and is compatible with a range of missions.
- *Mission Plan*: This item represents the operations of an individual UAV or a fleet of UAVs to fulfill the mission requirement.

The data schema can be instantiated for individual elements of a UAS. For example, Figure 6.8 shows the instances of the UAV, autopilot controller, and a payload sensor built for the case study presented in Chapter 5.

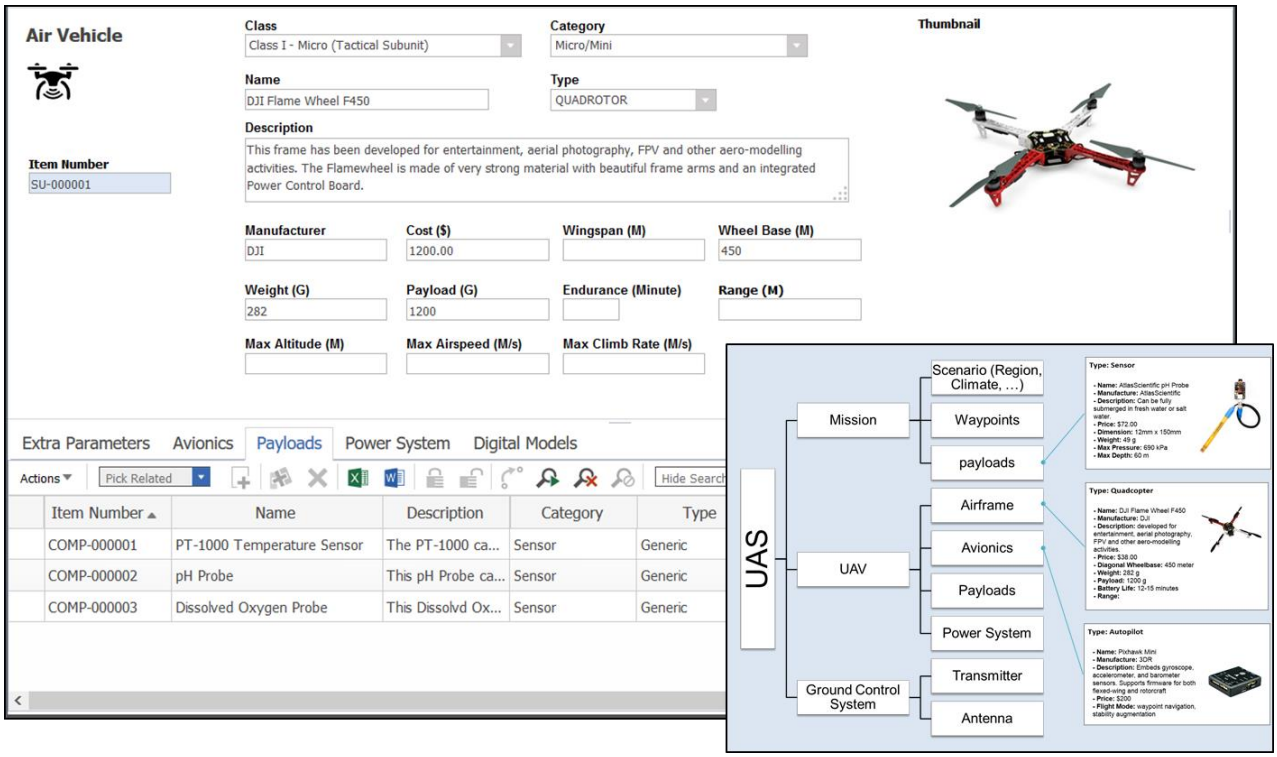


Figure 6.8 The instances of UAV, autopilot controller, and payload sensors

6.3.4 Ground Control Station (GCS) Integration

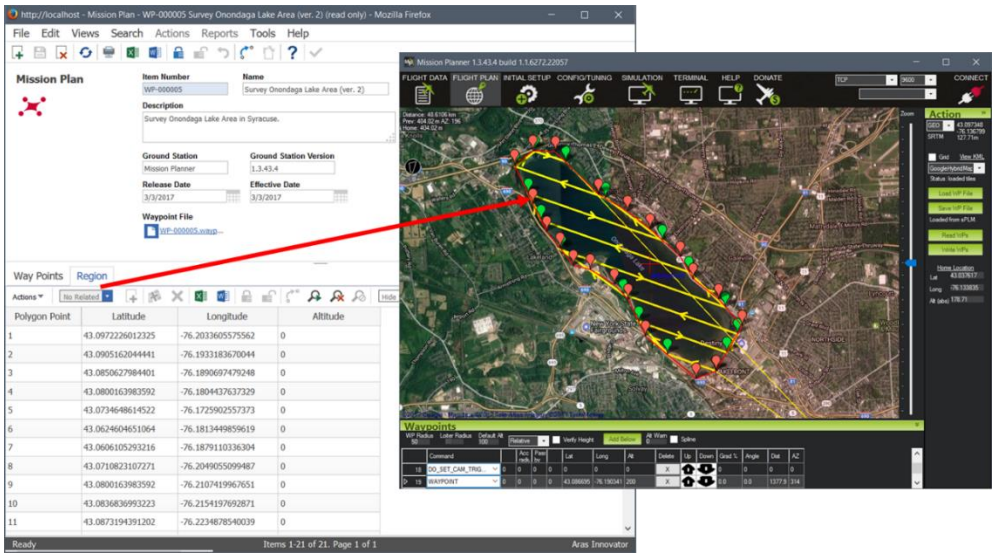


Figure 6.9 Save/load mission waypoints from sPLM

The ArduPilot Mission Planner is one of the popular, open-source, and full-featured ground station control applications that are used for path planning, controlling and monitoring the UAV. It is compatible with various types of vehicles: plane, copter, and rover. An sPLM connector was developed using the C# programming language to interface the mission planner software with the sPLM back-end server. This enables a remote pilot to be able to access a secure data/model repository for saving and reusing his mission plans (Figure 6.9), referencing a large set of shared information including the UAV and payloads specification data, as well as following a standardized workflow to collaborate with other stakeholders involved in the mission. Take the mission plan planning as an example, the sPLM connector allows to:

- Load/save polygon point coordinates from/to the sPLM server, in order to define the mission region;
- Load/save waypoints from/to the sPLM server, in order for analyzing, reusing, and tracing the mission path;
- Combine predefined regions and waypoints to create a new mission;

- Utilizing other UAS data (e.g. the UAV parameters, payloads parameters) from the sPLM server to generate regions and waypoints using custom algorithms.

6.3.5 Modular Design for Autonomy

It is important to design a modular UAS architecture so that the UAS platform can be easily reconfigured for various mission applications. Compared to the modularity of the physical components of a UAS, however, the modularity of autonomy is harder to achieve because the autonomy implementation is tightly coupled with other components and the data generated from the system. Below we use two examples to illustrate how the sPLM modeling framework helps the modular design of UAS autonomy.

Example 1: *State Estimate*

Obstacle avoidance is one of the data-intensive functions. It often follows a modular *Sense-Detect-Avoid* decision-making process: surveillance, state estimate and projection, conflict risk assessment, determination of appropriate avoidance maneuver, realization of the avoidance maneuver, and return to course (Lacher et al., 2007). State estimate is a critical predictive function for situational awareness.

One common approach for the state estimate module is to use an onboard inertial measurement unit (IMU) that processes observations from GPS and a set of sensors (gyroscope, accelerometer, magnetometer, barometer, etc.). These measurements are then fused by a set of algorithms to estimate/predict translational states (*position* and *velocity*) and rotational states (*attitude* and *attitude rate*). The translational states include north position (P_n), east position (P_e), altitude (h), north inertial velocity (V_n), east inertial velocity (V_e), and wind-relative airspeed (V_{air}). The rotational states include three Euler angles: yaw (ψ), pitch (θ), and roll (ϕ), as well as three angular rates ω_x, ω_y, and ω_z.

Several configurations of the IMU have been used on a UAS. Figure 6.10 shows a configuration

based on an attitude and heading reference system (AHRS) architecture, and Figure 6.11 shows a configuration based on a complete inertial navigation system (INS) architecture. The detailed mathematical formulations for AHRS and INS can be seen in Barton (2010).

In the AHRS architecture, the attitude angles are estimated with the aid of an IMU, and translational states are acquired directly from GPS and pressure sensors. It uses stable low-bandwidth attitude observations to estimate biases in the high-bandwidth angular rate gyroscope, then integrate the de-biased gyroscope measurements to form a complete attitude estimate. The INS architecture is a more computationally complex option for a UAS state estimation that combines the IMU and GPS measurements into a complete inertial navigation system. The low-pass-filtered translational state measurements from GPS are augmented with the higher-bandwidth acceleration on body rate measurements from the IMU. This is the main distinction between INS and AHRS. And an INS is typically implemented using an extended Kalman filter (EKF).

Both state estimation architectures can be represented as DMN notations that take the sensors data as inputs and generate state estimates (the decisions). While they have the same inputs and outputs, the architectures are different, so one interesting question is: which one is more modular?

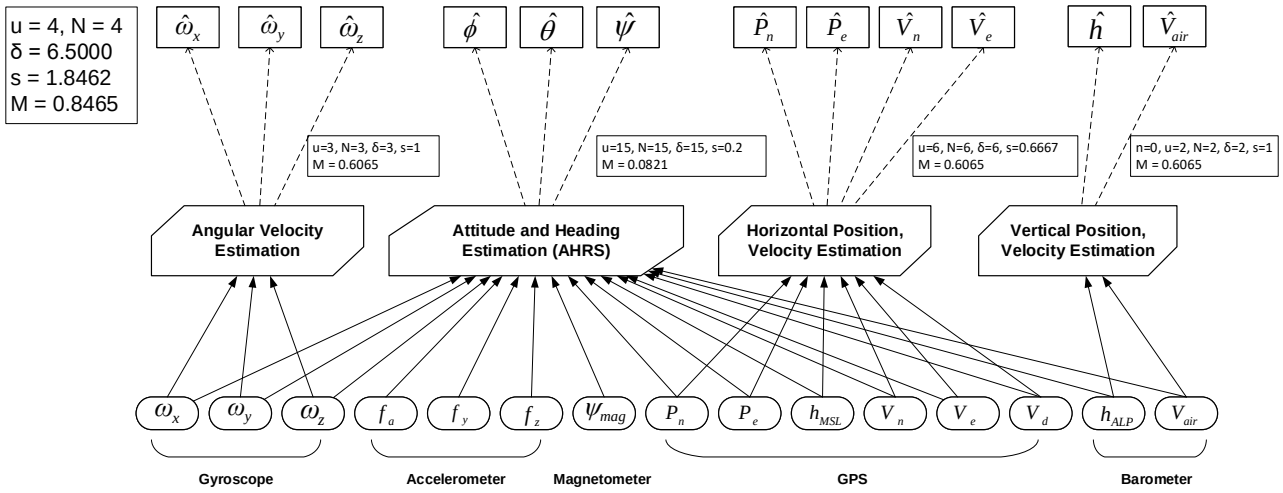


Figure 6.10 State Estimation using AHRS configuration (Li et al., 2017a)

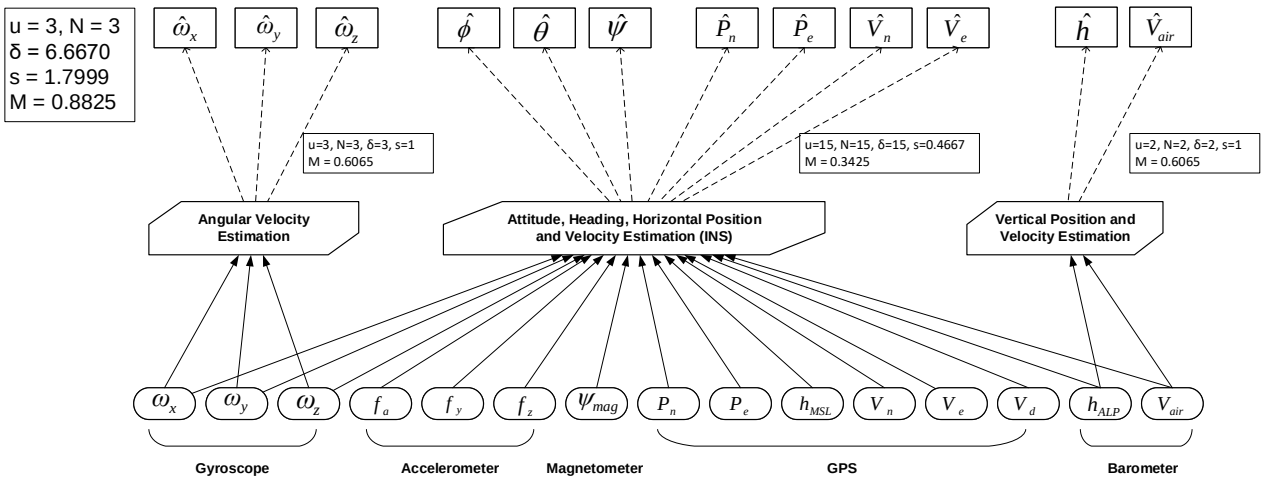


Figure 6.11 State Estimation using INS configuration (Li et al., 2017a)

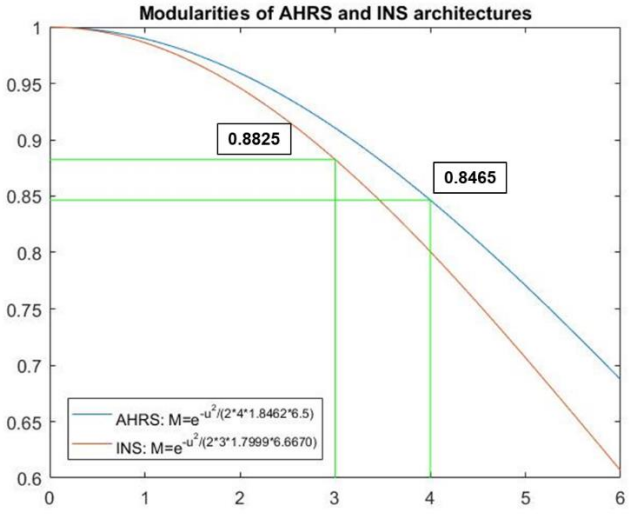


Figure 6.12 Modularity: AHRS vs. INS architectures

We can employ the analytics model modularity Equation 3.3 (Chapter 3) to evaluate the two architecture designs. Taking the INS architecture as an example, it has three unique modules, u=3 and N=3. It has 3+15+2=20 incoming links in total, the degree of coupling $\delta = 20/3 = 6.6670$. It predicts 12 decisions, the substitution factor $s = 12/(20/3) = 1.7999$. With all these numbers, $M_{AM,INS} = e^{-u^2/2Ns\delta} = 0.8825$. Similarly, the modularity for the AHRS architecture $M_{AM,AHRS} = 0.8465$.

The modularity for each sub modules can be computed in the same way. The modularity of the INS

architecture seems to be higher than that of the AHRS architecture due to the high modularity value of the INS module. For both architectures, the modularity will increase if more sub modules are implemented as standard analytics models (Figure 6.12). For instance, if the Angular Velocity Estimation and Vertical Position and Velocity Estimation modules can be seen as standard models (because they can be used by both the two architectures), the modularity for the INS and AHRS architectures become 0.9692 and 0.9349, respectively.

Example 2: *Path Planning*

Now, let's look at a more complicated autonomy function. Path planning is one of the main functions of the guidance system that includes: trajectory generation, path planning, mission planning, decision making, as well as reasoning and cognizance. Path planning is a process of using accumulated navigation data and *a priori* information to allow the UAV to find the best and safest way to reach a goal position or to accomplish a specific task. The decision logic of a path planning architecture is shown in Figure 6.13.

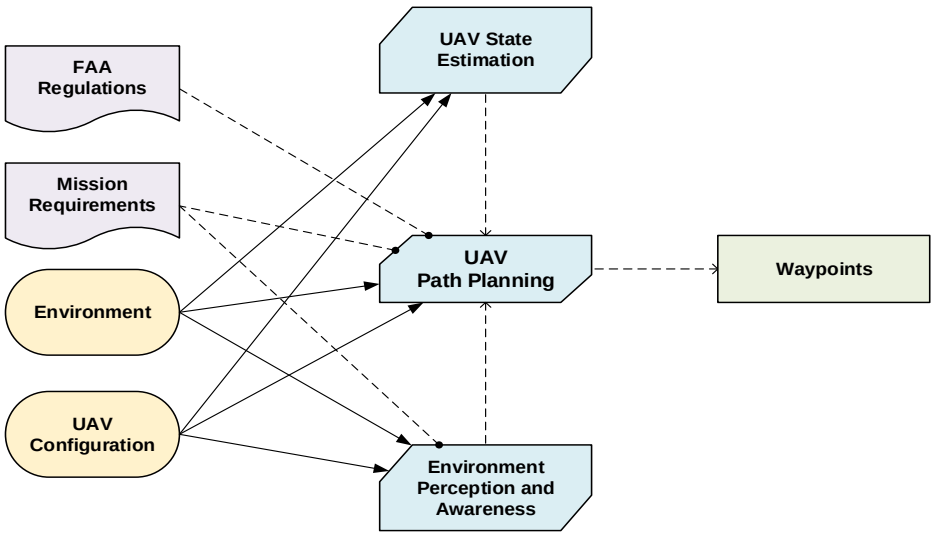


Figure 6.13 A Path Planning decision model (adapted from Li et al., 2017a)

With this modular architecture design, the path planning module can be substituted with appropriate

algorithms in different UAS applications. In an application as shown in Figure 6.14a, we use a fixed-wing plane carrying a high-resolution camera to survey a local lake. The path planning is done by a *Grid* algorithm to ensure the plane flying in the shortest path while covering the whole lake area. In the application as shown in Figure 6.14b, we use a quadcopter to explore the university campus with the campus map and building occupancy being given. The path planning is done by a sampling-based planner (SBP) algorithm, *Rapidly-exploring Random Tree Star (RRT*)*, to dynamically generate an obstacle-free path within the given map.

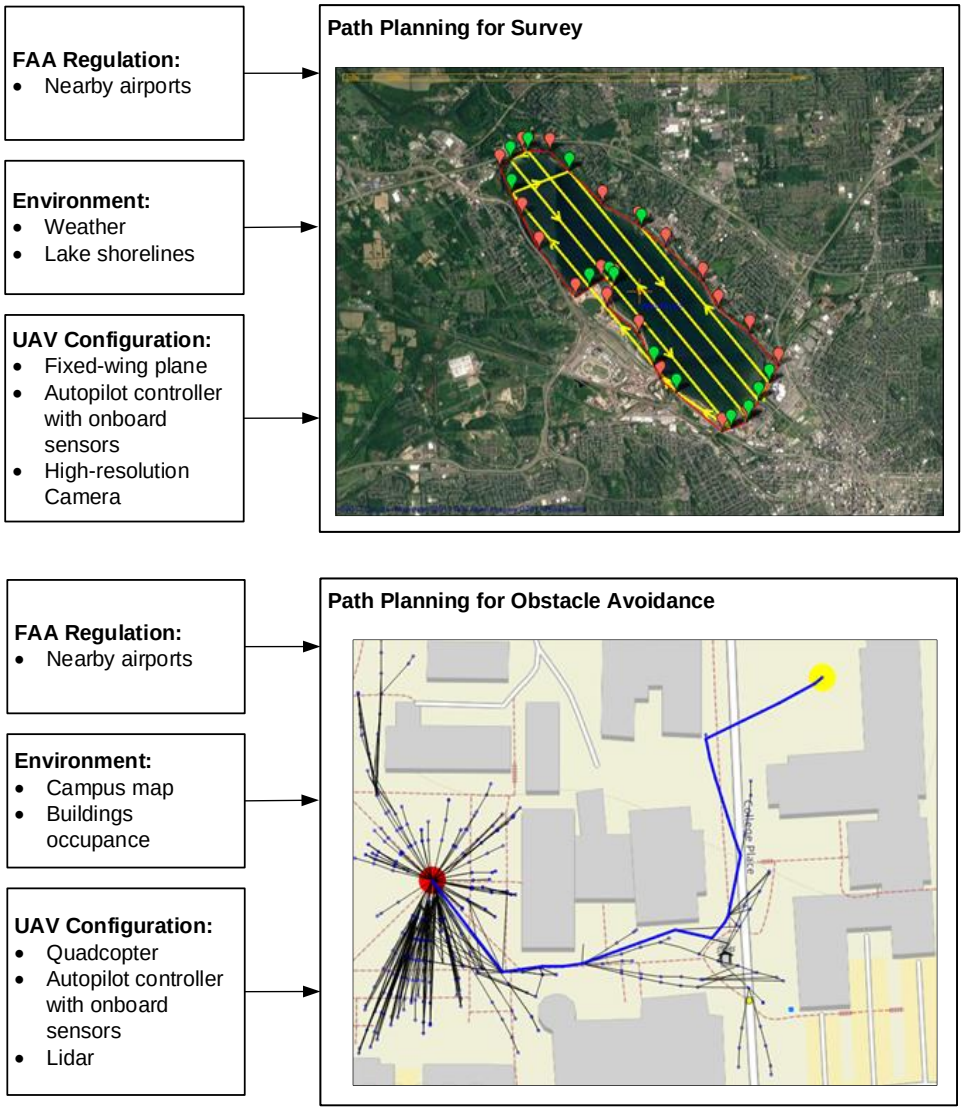


Figure 6.14 Modular path planning (Li et al., 2017a): (a) a lake survey application; (b) a campus exploration application

RRT* is an optimal version of the *Rapidly-exploring Random Tree (RRT)* algorithm, which is one of the most influential sampling-based motion planning algorithms. A sampling-based planner has been designed as a modular architecture, see Figure 6.15. Here, the concept of *C-Space* is used to simplify complex planning scenarios in the workspace of a UAV. Free space, C_{free}, and obstacle space, C_{obs}, are the two regions within the C-space, C. This prevents the need to explicitly define obstacles. The UAV can be only represented by a configuration, q, at any instance. A sequence of consecutively connected configurations represents a path, P. Start, q_{start}, and goal, q_{goal}, configurations are the inputs to the motion planner. The problem is to find a collison free path, P_{free}, which connects q_{start} to q_{goal}. A path is considered free if its entire configurations lie in C_{free} and their connecting paths do not intersect C_{obs}.

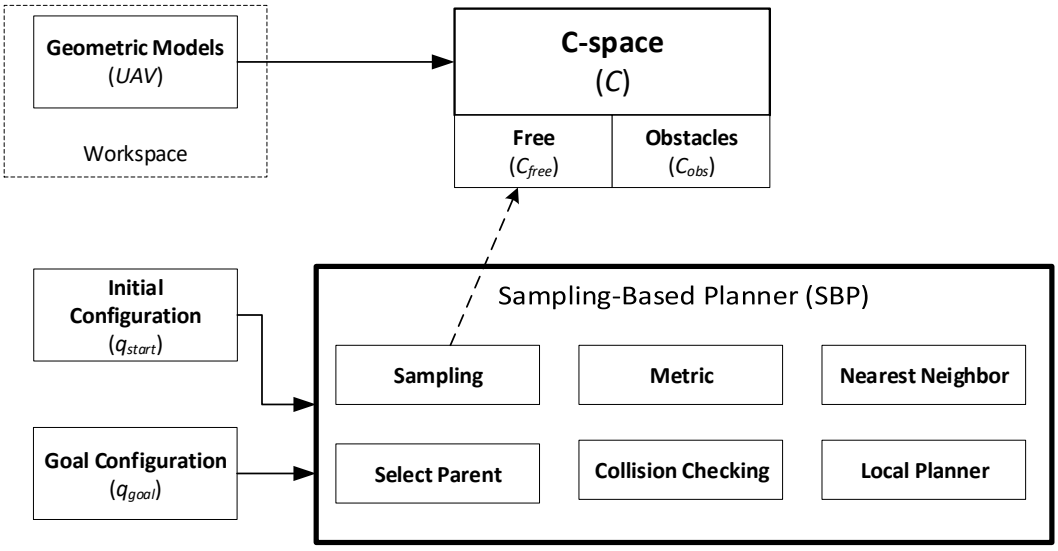


Figure 6.15 A general Sampling-Based Planner (adapted from Elbanhawi and Simic, 2014)

A sampling-based planner includes the below primitives:

- *Sampling*: This procedure is used to select a configuration, randomly, or quasi-randomly, and add it to the tree or roadmap.
- *Metric*: Given two configurations q_a and q_b, this procedure returns a cost value that signifies the effort required to reach from one configuration to another configuration.
- *Nearest Neighbor*: This is a search algorithm that returns the closest nodes to the new sample.

- *Select Parent*: This procedure selects an existing node to connect to the newly sampled node. Specifically, RRT selects the nearest nodes within its neighborhood.
- *Collision Checking*: This is a Boolean function that detects whether it does intersect with C_{obs} when connecting two configurations then returns a success or failure flag.
- *Local Planning*: Given two configurations q_a and q_b, this procedure establishes a connection considering kinematic or dynamic constraints.

The RRT algorithm has been shown probabilistically complete (Kuffner and LaValle, 2000) with an exponential rate of decay for the probability of failure. The typical procedure of an RRT algorithm is shown as below, in which (particularly the step 4) an RRT* considers all nodes in a neighborhood of a new sample and evaluates the cost of choosing each as the parent (Karaman et al., 2011):

1) The search is initialized from q_{start}.
2) A node, q_{rand}, is selected from the C-space using the sample procedure.
3) q_{rand} is discarded, if it is in C_{obs}.
4) The nearest neighbors are searched and q_{near} is returned according to the metric.
5) The local planner is used to connect q_{rand} and q_{near}. The planner may return q_{new} that $q_{r}$and may not be reachable. If q_{rand} is not reached, it is discarded.
6) Collision checking is performed to ensure the path between q_{near} and q_{new} is collision free. If path is collision free, q_{new} is added to the tree.
7) The search terminates when q_{new} equals to q_{goal}, a number of iterations is exceeded, or a specified time period is exceeded.

In sPLM, the algorithm settings or analytics model settings can be stored in the database for each mission instance. The algorithms can automatically take the mission settings (map, geofence) and hardware data (camera parameters, flight speed) into their computations. For the second case shown in Figure 6.14, we also process the input map to a binary image with black pixels standing for obstacles in order to separate the free space and obstacle space. Other obstacle data (e.g. the tower data from the public or government database) can be imposed in a similar way if needed. Finally, the path generated by the algorithm needs to be transformed to waypoints based on the World Geodetic System 1984

(WGS84), see Figure 6.16. The modular architecture allows other custom algorithms to be integrated into the sPLM platform for automation.

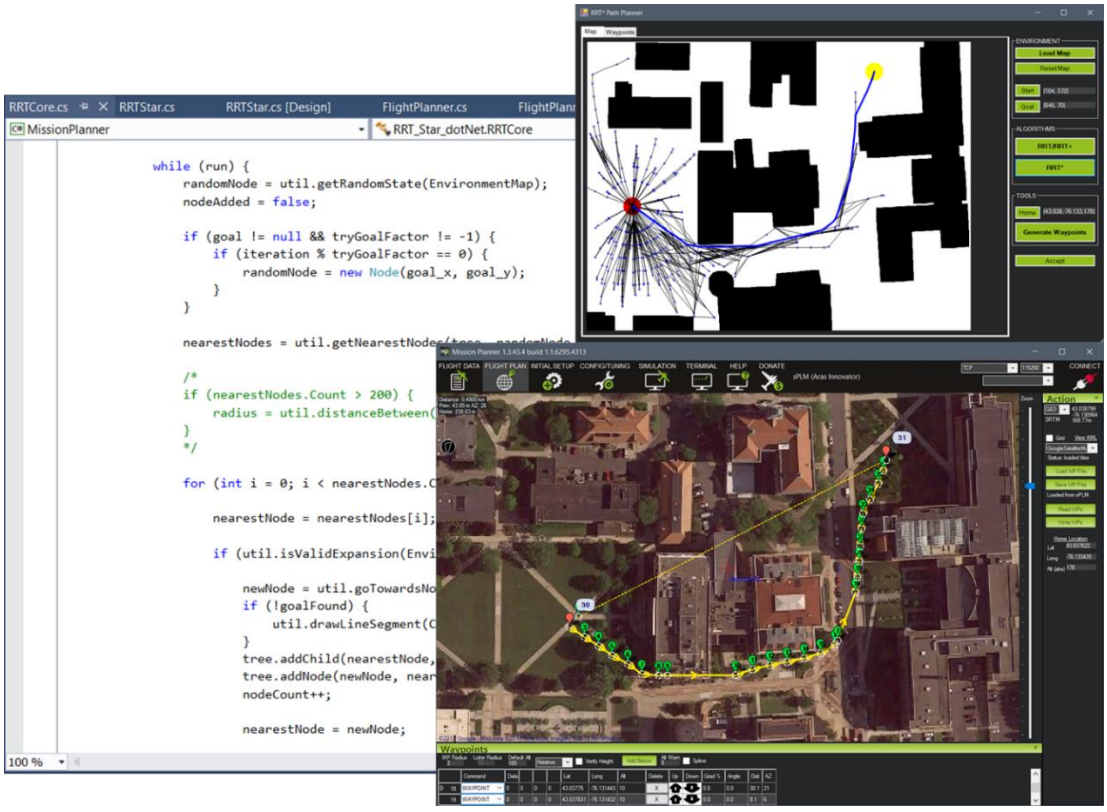


Figure 6.16 Using RRT* algorithm in sPLM for path planning[28]

6.3.6 PLM-In-The-Loop (PITL) Simulation

The sPLM as a backbone can also be taken into the UAS modeling and simulation loop, this is another advantage to help managing the autonomy. As shown in Figure 6.17, there can be three levels of capabilities to take the PLM in the loop (PITL) of a UAS, depending upon where the intelligence will be implemented:

- *Level 1*: The sPLM server acts as a data repository for the UAV, payloads, and mission data. The decision will be made in the ground control station (e.g. Mission Planner) or specific

[28] **Note:** the C# code used in this example is adapted from the java code developed by the Correll Lab at University of Colorado: http://correll.cs.colorado.edu/?p=1623.

computational platform (e.g. MATLAB).

- *Level 2*: The sPLM server also acts as a model repository to host necessary models (digital representations of physical components or analytical components) related to a UAS. The ground control station can work as a client to compose/configure the models from the sPLM to fulfill a specific mission.
- *Level 3*: The sPLM platform provides various services for clients ranged from mission planners to various elements of the UAS. For instance, the onboard computer can also send/receive data to/from the sPLM platform.

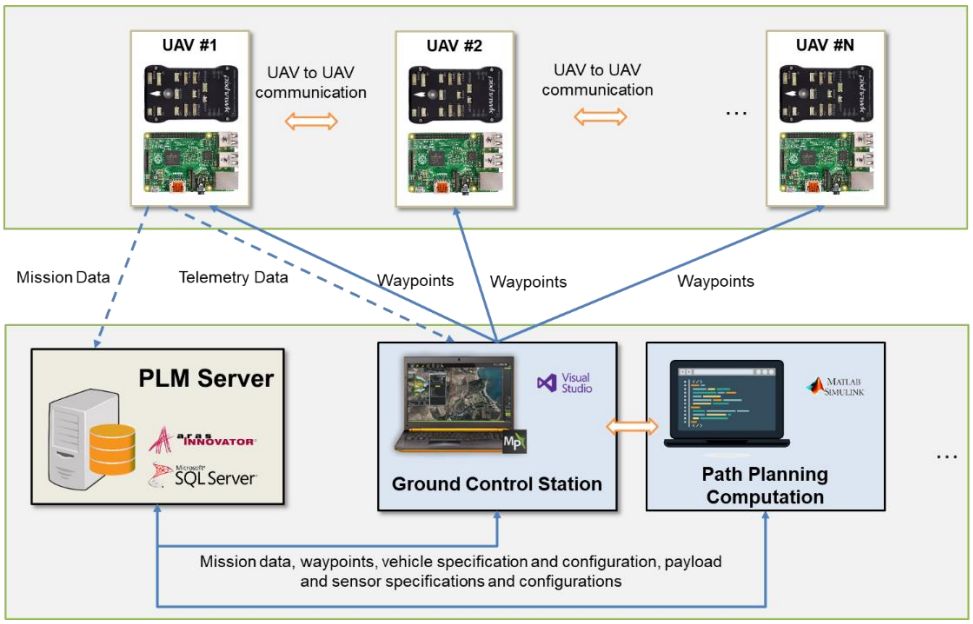


Figure 6.17 PLM-in-the-loop UAS Architecture

Table 6.1 Four UAV configurations

UAV #1	UAV #2	UAV #3	UAV #4
TCP/IP: 127.0.0.1 Port: **5760** Firmware: ArduCopter Type: Quadcopter (QUAD) Home: [43.037623, -76.133434,116] Speedup: 3X	TCP/IP: 127.0.0.1 Port: **5770** Firmware: ArduCopter Type: Hexacopter (HEXA) Home: [43.037776, -76.133200,116] Speedup: 3X	TCP/IP: 127.0.0.1 Port: **5780** Firmware: ArduCopter Type: Quadcopter (QUAD) Home: [43.037621, -76.133200,116] Speedup: 3X	TCP/IP: 127.0.0.1 Port: **5790** Firmware: ArduCopter Type: Hexacopter (HEXA) Home: [43.037458, -76.133200,116] Speedup: 3X

For example, to support a four-UAV mission simulation, the individual UAV parameters shown in Table 6.1 are retrieved from the sPLM server to initialize four UAV simulator instances (the base

simulator is also stored in the sPLM server). Each UAV has its unique system ID (**SYSID_THISMAV**) and other parameters conforming with the micro air vehicle communication protocol (MAVLink[29]). A multi-connection can be established between the Mission Planner and the simulators, then coordinated control commands can be sent to individual UAV. The path planning for the four UAVs is based on a leader-follower swarm algorithm, but other path planning algorithms can be used as well. Figure 6.18 shows the simulation and the real-time telemetry of the four-UAV formation mission.

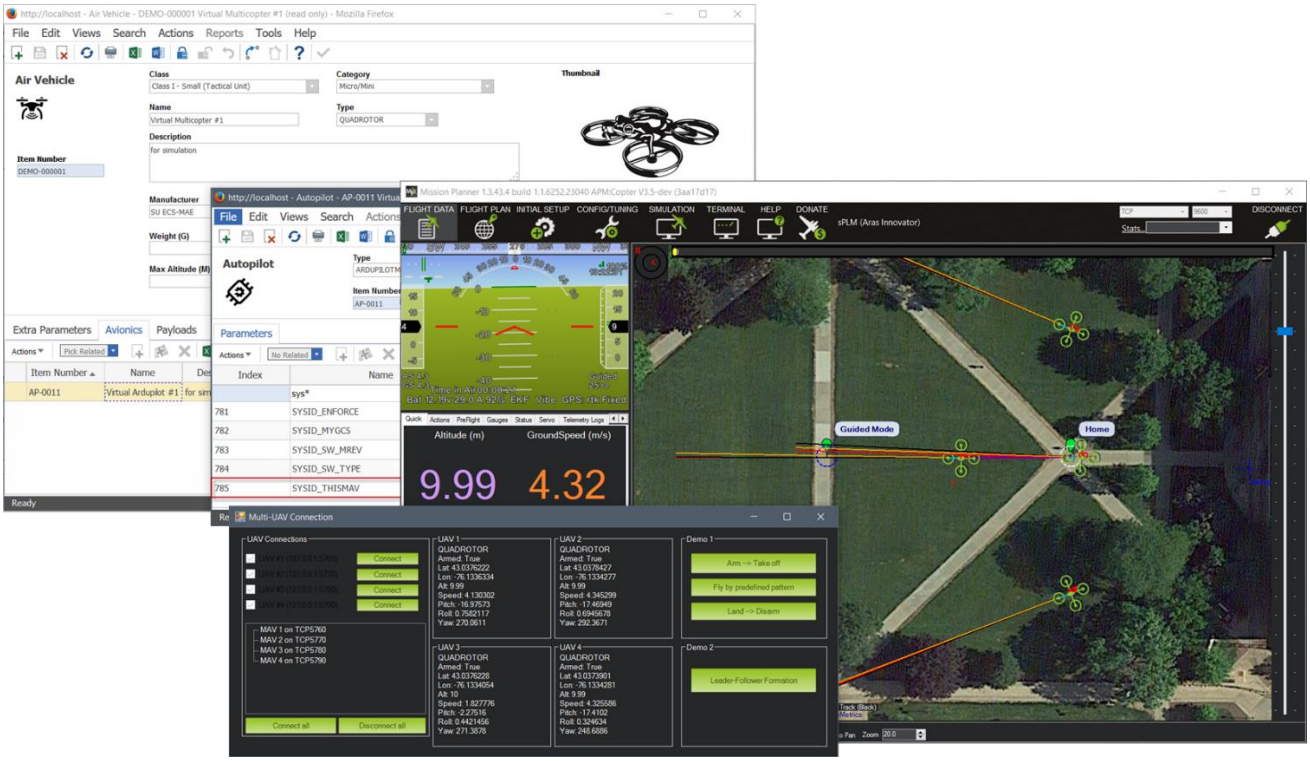


Figure 6.18 Multi-UAV flight simulation

6.3.7 Mission Traceability and Forensic Analysis

All the data, models, and configuration relationships involved in the UAS application development and operations are recorded in the sPLM database for monitoring and tracing. For instance, the bill of data and bill of materials shown in Figure 6.19 is for the completed mission of the campus exploration.

[29] MAVLink Developer Guide, https://mavlink.io/en/

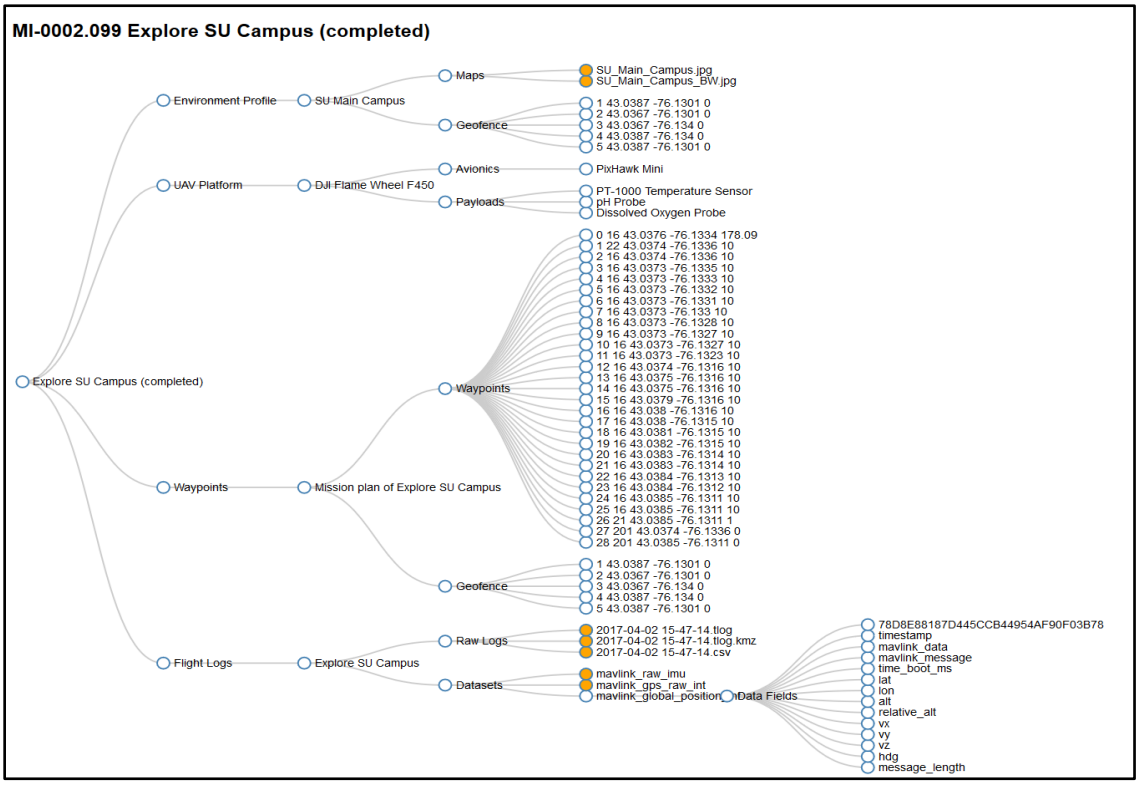


Figure 6.19 The bill of data and materials of a UAS mission (Li et al., 2017a)

The tracing of these data help to answer common questions in mission and incident investigation (Kovar and Bollo, 2018):

- *What happened during this flight? Where did it start, how high did it fly, what route did it take?*
- *What other flights did this aircraft perform? What other sites might have seen this aircraft?*
- *What is the history, flight, maintenance, software, and firmware, of this aircraft?*
- *Was an expert who maintains it capable of modifying the firmware or hardware?*
- *What components of the aircraft are uniquely identifiable and traceable?*
- *What identifiable components, such as batteries, are shared with other aircraft? Can we link this aircraft to a larger operation?*
- *What other devices, services, individuals and accounts are related to this aircraft and how can we identify them? how do we reach out into social media, third party data services, and the physical world using the data on the UAV?*

For example, during a flight test (in July 2018) using an Intel Aero quadcopter, the UAV ran out of control and crashed, one propeller and the GPS pole stand were broken. The ground control system

warned that the communication link was lost (this UAV has a maximal remote-control distance up to 300 meters). In order to investigate the cause of the accident, a data analytics pipeline was built on the sPLM server to connect the UAV flight log data, the UAV mechanical specification data, and other mission-relevant data. This pipeline can automatically process the log raw data into reusable data packages ready for further analysis and visualization. Figure 6.20 shows the key steps and intermediate results of this data processing pipeline.

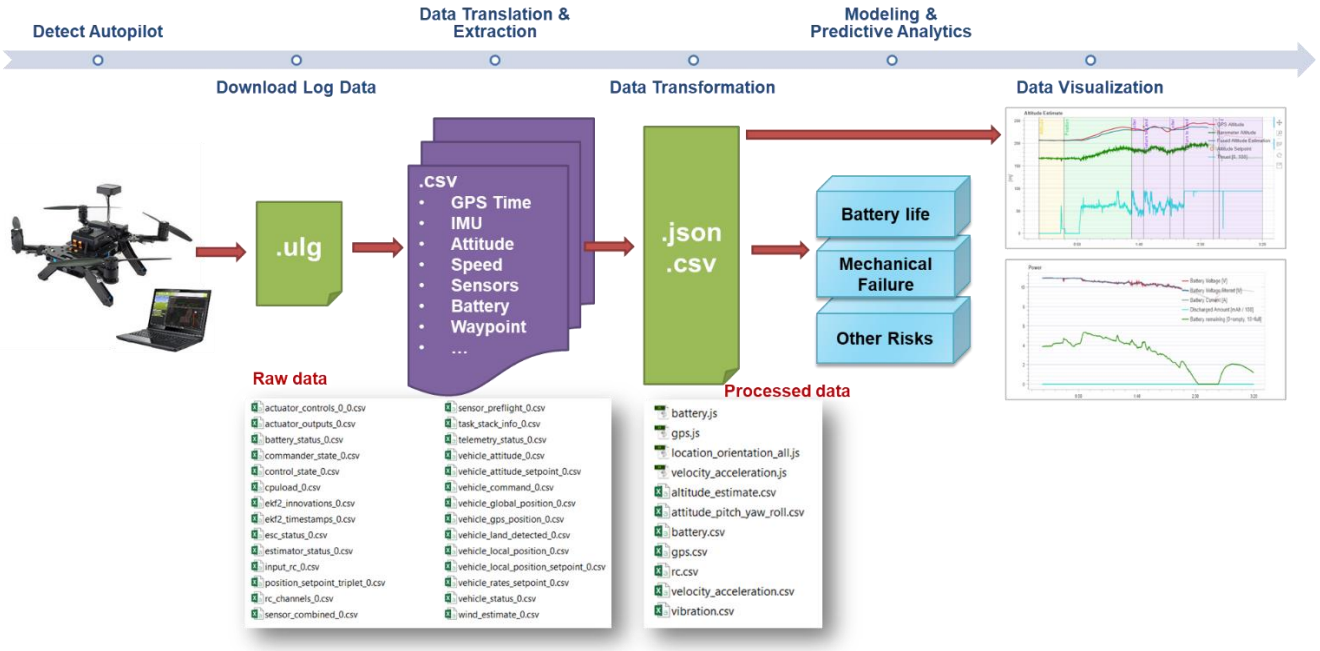


Figure 6.20 The log data analytics pipeline for the Intel Aero UAV

First, the onboard log data was downloaded from the UAV. The raw data was initially encoded as *.ulg* file format so our next step was to convert it into a set of .csv files using the *pyulog* toolkit[30]. The processed csv files were then aggregated and categorized into nine data packages: *Position*, *Orientation, Velocity*, *Navigation*, *Power*, *Sensors*, *Payloads*, *Remote Control*, and *Event*. Each data package was time tagged and aligned.

[30] pyulog on GitHub, https://github.com/PX4/pyulog

Two types of time offsets were also calculated: (1) *boot time offset*: it is the offset between the boot time and the data point time; and (2) *take-off time offset,* it is offset between the takeoff time and the data point time. Figure 6.21 shows the code snippet (in R programming language) to calculate the altitude data offset time using the GPS time as reference. Figure 6.21 also shows the processed log data (encoded in JSON) ready for visualization in a web-based 3D virtual earth engine, Cesium (Appendix B).

```r
# determine drone boot time
# compare "timestamp" and "time_utc_usec"
dataFile <- 'raw/vehicle_global_position_0.csv'
rawData <- read.csv(dataFile, header=TRUE)
absoluteTimeInSeconds <- rawData[1,]$time_utc_usec/1e6
relativeTimeInSeconds <- rawData[1,]$timestamp/1e6
bootTimeInSeconds <- absoluteTimeInSeconds - relativeTimeInSeconds
bootTimeUTC <- as.POSIXlt(bootTimeInSeconds, origin="1970-1-1", tz="GMT")


# Altitude
dataFile <- 'raw/vehicle_global_position_0.csv'
rawData <- read.csv(dataFile, header=TRUE)
procData <- rawData[,c('timestamp', 'alt', 'pressure_alt')]
procData$bootOffsetTimeInSeconds <- procData$timestamp/1e6
procData$startOffsetTimeInSeconds <- (procData$timestamp-procData[1,]$timestamp)/1e6
procData$timestamp <- as.POSIXlt(bootTimeInSeconds + procData$timestamp/1e6, origin="1970-1-1", tz="GMT")
procData$index <- as.numeric(rownames(procData))
procData <- procData[,c('index', 'timestamp', 'bootOffsetTimeInSeconds', 'startOffsetTimeInSeconds',
                        'alt', 'pressure_alt')]
exportCsvFile <- 'processed/altitude_estimate.csv'
write.csv(procData, file = exportCsvFile, row.names=FALSE)
```

```
var logTimeDomain = {
boot : "2018-07-20T18:44:36.220962+00:00",
start: "2018-07-20T18:44:54.999954+00:00",
stop : "2018-07-20T18:47:56.215776+00:00"
};

var logHomePosition = {
longitude : -76.1127606000,
latitude  : 43.0078997000,
altitude  : 206.989
};

var droneModel = "/apps/usplm/SampleData/Resources/models/Intel_Aero_R1L.glb";

var logFlightMode = [
["2018-07-20T18:44:55.230146+00:00", "Manual"],
["2018-07-20T18:44:55.315126+00:00", "Altitude"],
["2018-07-20T18:45:15.620111+00:00", "Position"],
...
];

var logPosition = [
["2018-07-20T18:44:54.999954+00:00", -76.1127606000, 43.0078997000, 206.989],
["2018-07-20T18:44:55.207864+00:00", -76.1127604000, 43.0078993000, 207.011],
…
];


var logOrientation = [
["2018-07-20T18:44:54.900759+00:00", 0.010801, 0.021432, -0.252918, 0.967190],
["2018-07-20T18:44:54.993386+00:00", 0.010785, 0.021333, -0.254711, 0.966722],
…
];

var logEvent = [
["2018-07-20T18:46:37.900759+00:00", "[WARNING] ", "Manual Control Lost. Failsafe Enabled: No RC."],
["2018-07-20T18:46:46.900759+00:00", "[WARNING] ", "All Data Links Lost."],
…
];
```

Figure 6.21 A snippet of R code to calculate the offset time and the processed log data in JSON format

The mission start time and end time were extracted to define the time domain of the mission. The location (*longitude*, *latitude*, *altitude*) and orientation (*pitch*, *yaw*, *roll*) data were encoded as time-dynamic properties of the Cesium entities to animate the UAV attitude using 3D models. Other telemetry data (including event data) was also encoded as corresponding Cesium entities. The UAV CAD models (the digital representation of physical components) were translated into glTF format and were then rendered in the Cesium engine. The environment data (map, terrain, weather, airspace, buildings, air traffic, etc.) was transformed into geospatial data and overlaid on the Cesium virtual Earth. The final result is shown in Figure 6.22.

Figure 6.22 Forensic analysis for the Intel Aero UAV failure (image courtesy: UsPLM, Inc.[31])

The critical events extracted from the log are shown in Table 6.2. These messages and the flight

[31] UsPLM, a drone fleet management software company. www.usplm.net

animation in Cesium reveal a chain of accidents, where (1) the communication link lost caused the UAV to take the return-to-home action; however, (2) the loose installation of the battery caused instable movement of the UAV so it could not keep the right path and had to maneuver to offset the flight route, and this consumed more battery energy than expected; then (3) the inaccurate remaining battery life prediction from the onboard algorithm caused the UAV to take emergency landing procedures when it "thought" the battery had gone to the critical level (see Figure 6.23); and finally (4) the movement of the battery caused the UAV not be able to maintain the balance and touchdown the ground with one airframe arm first.

Table 6.2 The events extracted from the flight log

#	*Time*	*Level*	*Message*
0	0:01:43	WARNING	MANUAL CONTROL LOST (at t=103946ms)
1	0:01:43	WARNING	Failsafe enabled: no RC
2	0:01:52	WARNING	ALL DATA LINKS LOST
3	0:02:05	WARNING	ALL DATA LINKS LOST
4	0:02:16	WARNING	MANUAL CONTROL LOST (at t=136534ms)
5	0:02:16	WARNING	Failsafe enabled: no RC
6	0:02:23	WARNING	LOW BATTERY, RETURN TO LAND ADVISED
7	0:02:28	ERROR	CRITICAL BATTERY, RETURN TO LAUNCH ADVISED!
8	0:02:29	ERROR	DANGEROUS BATTERY LEVEL, LANDING ADVISED!
9	0:02:44	WARNING	MANUAL CONTROL LOST (at t=164996ms)
10	0:02:44	WARNING	Failsafe enabled: no RC

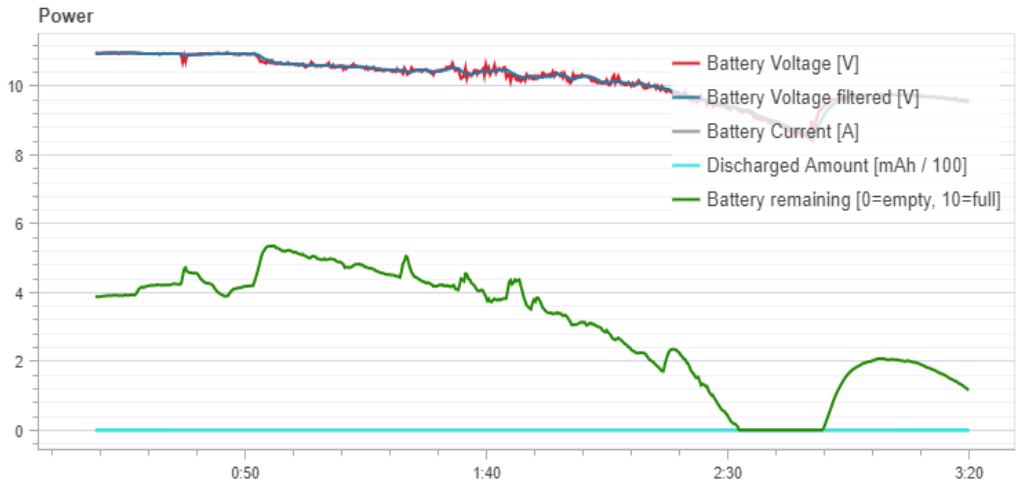


Figure 6.23 The battery level of the Intel Aero UAV during the test flight

This example shows how the sPLM platform does play an important role of data/model repository to enable UAS modeling and simulation in its utilization stage. In this way, the operational data and the analytics models for autonomy are treated as another types of assets so they orchestrate with the physical asset data to construct a full digital thread for connecting different components of the UAS and for tracing any part of the mission.

6.4 Validation of the sPLM Platform

The development of the UAS lifecycle management platform validated that the sPLM modeling framework is usable and extendable to build industry-specific applications. It was co-developed with the Smart UAS project discussed in Chapter 5 during December 2016 to April 2017. The sPLM platform was initially used to support the development of a Smart UAS for a Water-Quality-Sampling application requested by a civil engineering scientist. The sPLM platform provided the UAS development team a collaborative environment and data repository to facilitate effective data/information exchange, visual communication, and traceable decision-making. The integrated process model, NPD^3, presented in Chapter 5 also provided a common language and guidance to both engineers and data scientists involved in the project. They otherwise are not familiar with the process used by their counterparts.

To validate the sPLM concept in broader UAS industry, the author and his colleagues carried out three primary market survey projects during late February to early April, 2018, when the author had his practical trainings in the UsPLM, Inc. at Syracuse, New York, USA. The three surveys were based on focus group, video/audio interview, and online survey, respectively. In all three surveys, we showed the participants the UAS lifecycle management concepts described in Section 6.3.2 and then asked them a set of questions (the questionnaire is listed in Appendix D). Table 6.3 summarizes the brief description for each survey; Table 6.4 shows some feedbacks from the target audiences. Note we interchangeably

use the terms “UAV” and “drone” in these surveys.

The surveys reflect the lifecycle management challenges faced by the UAS fleet operations stakeholders, for which the sPLM framework can provide values. That is, the sPLM concept and its implementation for this specific industry are valid.

Table 6.3 Three types of surveys

Survey	*Date*	*Target Audience*	*Description*
Survey #1: Focus Group	Feb. 9, 2018	University Researchers	We invited seven professors who are using UAS for research in two local universities: SU and SUNY ESF. The professors are from various disciplines including mechanical engineering, electrical engineering, civil engineering, information management, and architecture. The goal is to understand the challenges faced by scientific researchers employing UAS for research.
Survey #2: Video/Audio Interview	Mar. 5-Mar. 30, 2018	Operations Managers in large companies or UAS service providers	We worked with a local leading market research company, recruiting ten participants for in-depth interviews, 45-60 mins in length each. The goal of this research is to (1) explore UAV fleet profiles and uses, (2) understand current challenges operating and managing UAV fleets, (3) explore market reactions to the solution concept, (3) capture reasons potential customers like or dislike the solution (concept review), (4) receive inputs on pricing options for the solution, and (5) determine likelihood to purchase as well as any potential barriers to purchase.
Survey #3: Online Survey	Mar. 27-Apr. 6, 2018	Small UAS Remote Pilots	We partnered with a local FAA certified training service provider, to survey a base of 700+ small UAS operations professionals and we got 60+ responses in this survey. The goal is to understand the challenges faced by the frontline UAV operators.

Table 6.4 Survey feedbacks

Survey	*Feedback*
Survey #1	“Currently, we do a lot of what you are proposing to include in your software by hand/by foot (mission planning, checking for obstacles, reading weather reports) - If you could speed that up and make it ‘automatic’, that will make not only my life easier, but also the lives of other scientists interested in these platforms easier as well!” – A professor from the Department of Earth Science, Syracuse University
Survey #2	“[…] mission management, just kind of brings my attention to ... Yes. Because I like what I am reading here that, mission management and we are all on a mission and we all have some goals and some types of mission to achieve. […] it would be very nice to have with a company to deliver that comprehensive.” “User management right off the top. Very useful as well […]. UAV and payload management for managing the advised asset data and maintenance history, includes asset data […] I would like to be incorporated as well. Very useful indeed. Doesn't need too much explanation.” “I want mission management. I want API connectors. And I want UAV and payload management. I

	like autonomy model management as well. […] Because it includes algorithm for task planning what is tremendously important. Obstacle avoidance is tremendously important. Data analytics, machine learning models, Yes, it's in the raw stage, yet at the beginning stage of machine learning and AI. But still we are going there. That's the future of our industry." – A manager in a large company using UAS for warehouse surveying and item delivery
Survey #3	"There are an overwhelming number of apps to choose from. I stick with the tools and apps I'm comfortable with." "Some of the tools have been hard to use in areas where cell phones do not function as information is not recorded accurately." "I expect a few hours of configuration before getting the tools to do what I want." "I've missed opportunities to shoot things because I was updating firmware and being careful." "The customized drones that we are using are not very robust, it is hard to identify problems and not to mention how to resolve problems. It also makes the simulation unreliable and unpractical."

Chapter 7

Conclusion & Closing Remarks

Autonomy and intelligence have been built into many of today's mechatronic products, taking advantage of low-cost sensor technologies and advanced machine learning algorithms. Design of product intelligence is no longer a trivial or additional option for the product development; instead, it calls for a systematic approach to understand the components and their characteristics that contribute to the intelligence. We argued that a smart product can be seen as an appropriate configuration of the physical components to form its body and the analytics models that implement its intelligence functions. Thought of this way, creating smart products involves in co-development of the two components in a multi-disciplinary design space. We then contextualized this design space as a **Smart Products Hypercube** that projects the physical components design, analytics models design, and time-based lifecycle stages on three orthogonal dimensions. The decomposition of this high-dimensional space formulated the research domains (A-T Space, P-A Space, and P-A-T Space) that can contribute to developing the next generation PLM system for smart products development.

To start with, our first question was: Can an analytics model be treated as a product? To prove this, we conducted a systematic review on the theoretical framework regarding product and process modeling practices for physical products and then investigated the issues that are related to the current practice to develop complex analytics models. The standardization of the model family architecture and individual model structure, the possible modular design approaches, the modularity quantification, and their impacts on the overall smart product design were studied systematically. We found that analytics models possess both product characteristics and decision-making characteristics. On the product perspective, each analytics model can be seen as a result of the production of chunks of data; data is the "material" to

construct the analytics model in a certain kind of form. On the decision-making perspective, an analytics model takes data as inputs and generates decisions. A data object itself is also an analytics model that directly outputs the input as a decision to the next analytics models to form a decision chain. Both perspectives suggest that data must be a constituent of an analytics model. And both perspectives imply it is feasible to translate the conventional task-oriented data analytics activities into a formal data product engineering process.

The next question was, can we apply (or adapt) the methodologies existing in traditional manufacturing domain to the data products (analytics models)? Two product family design approaches, scale-based and module-based approaches, had been adapted to develop parametric and combinatorial analytics models. A modularity function had also been derived to evaluate the degree of modularity of an analytics model architecture. These concepts to model analytics models as products were then implemented in an open-source PLM platform to support development of data analytics models for a smart CNC machining center. The implementation proves the feasibility to apply the PLM concept for analytics model lifecycle management. Therefore, we provided a feasible solution to the **A-T Space problem**.

Next, we postulated a higher-level abstraction of information model that would be able to harmonize the standard information models already established and accepted in manufacturing and data science communities. We believe such a model can uniformly represent the information of physical components and analytics models so they can be composed at different levels (data, unit model, component model, and product). To prove this, we compared different elements involved in the development of physical components and analytics models. We found that they share commonalities in many aspects along their lifecycle stages. This provides the foundation to develop a **Smart Component** data model to consistently compose both models at the same time as a uniform, hybrid model. The

analytics model lifecycle management established in the first stage enabled us to take advantage of the PLM capability that is traditionally designed for physical product development. Thus, the smart component data model enables engineers, data scientists, and other stakeholders to collaborate on a common platform to develop smart products. It also serves as the basic foundation for building smart devices, smart equipment, and smart services, which are the key components of a smart manufacturing system. The smart component can also be seen as a special implementation of the I4.0 Component built for Industry 4.0 architecture, meaning our model can be possibly applied in more broader manufacturing applications.

Three well-adopted standards – the ISO STEP, OMG DMN, and DMG PMML – were employed to capture the product-related data/information as well as to decompose the decision logics of a system including physical components and analytics models. We validated the smart component model by using it to develop a modular, hybrid sustainability assessment system where both knowledge-based models and data-driven models can be accommodated for and combined at different abstract levels in assessing a product's sustainability. This provided a possible solution to address the **P-A Space problem**.

Literatures had shown that misalignment of the product architecture and the development team organization may have a negative impact on product performance (Sosa et al., 2004). Therefore, we argued that mechanical and electrical engineers need to work closely with software engineers and data scientists to co-develop the components of a smart product. In order to develop an integrated process model to support such a transdisciplinary collaboration, we revisited the classic new product development (NPD) process model and a well-adopted data analytics process model, CRISP-DM, to understand the key tasks prescribed for the engineers in a physical product development team and the data scientists in a data products development team, respectively. We then developed an **NPD^3 Model** and evaluated it within a case study of the creation of a smart unmanned aircraft system.

The results of our case study demonstrated that there was cross-disciplinary design information required by the engineers as well as the data scientists, and that it was critical that direct interactions and messages were exchanged between the two groups, with a project management group acting as a mediator to guide the collaboration across the team. In addition, the project management group could help ensure that the needed external design information be shared between two disciplinary groups. It also showed that the timing and contents of information exchanged between the two groups facilitate the information awareness and access, knowledge transfer, and problem-solving. We developed a theoretical framework to characterize the interaction patterns. The NPD^3 model and the PLM implementation supplied the UAS development team a collaborative environment and data repository to facilitate effective data/information exchange, visual communication, and traceable decision-making. This study had been a starting point to address the **P-A-T Space problem**.

7.1 Innovation and Contribution

The innovation of this research is viewing data analytics as a **Data Products Development** process based on the recognition of how data analytics is contributing to the increasing intelligence of mechatronics. This opens a large opportunity to apply the established knowledge, theories, and practices in traditional manufacturing domain to data science. While data analytics produces software-like products, the data products are the results of new knowledge production (emerging knowledge) taking data as raw materials. This is contrary to regular software applications that have been built to automate repeatable human knowledge (i.e. known knowledge). Thought of this way, the author believes it is feasible and valuable to adapt traditional product development methodologies (e.g., modular design, mass customization, and product lifecycle management) across the data products development value chain (design, production, logistics, and supply chain), if this value chain exists or if it could be

established.

Seeing the commonalities between data products and physical products, the foundation of this research relies on the adaptation and extension of the "Product Lifecycle Management (PLM)" concept traditionally for physical products to data products. The goal is to integrate the long-term established theoretical frameworks and methodologies for traditional product development with the ongoing efforts of developing formal methodologies in the data science community. If this framework is successful, it would be greatly beneficial to both data science and manufacturing communities. While this research only addresses several critical problems (data/model interoperability, transdisciplinary collaboration, modular product architecture), the author believes it opens a door to explore a plethora of existing theories and methodologies to a new area, and eventually might merge them together.

7.2 Achievements

This research has addressed three problems regarding today's smart products development due to the adoption of the new discipline, data science, into product development processes. The objective of this research is to develop a formal modeling framework for smart products development involving co-development of physical components and data-driven features (enabled by analytics models). First, the framework had addressed the interoperability issues by developing a **Smart Component Data Model** to uniformly represent and compose physical component models created by engineers and analytics models created by data scientists. Second, the framework advocated an **NPD^3 Process Model** by integrating the generic new product development model with formal data analytics process model to support the transdisciplinary information flows and team interactions between engineers and data scientists during the concept development stage. Third, the framework had explored the feasibility of applying the theoretical framework in **Modular Product Design and Development** to address the issues related to

product architecture design, modularization, product configuration, and lifecycle management of analytics models.

A **Smart Products Lifecycle Management (sPLM)** proof-of-concept platform had been implemented for validating the concepts and methodologies proposed throughout the research. The sPLM platform is based on open-source tools and standards so that it provides a shared data repository to manage the product-, process-, and configuration-related knowledge for smart products development, as well as provides a collaborative environment to facilitate transdisciplinary collaboration between product engineers and data scientists.

The sPLM platform had been validated in a boarder perspective. Two types of smart products – one for consumer appliance (e.g. smart thermostat) and one for industrial equipment (e.g. CNC machines) – had been considered for scenario design, data collection, case analysis, and concept illustration. The sPLM platform was further adapted and specialized for the development and operations of unmanned aircraft systems, namely, UAS lifecycle management.

Since this research had utilized real industrial cases to develop solutions, the solutions and the methodologies to reach the solutions could be directly taken as reference examples for industrial practitioners, at least for the specific industries related to this research. The implementation had been based on industry-level open-source platform and tools; thus, it would provide an immediate toolset for users working on these tools. We expect the solution would contribute to the open-source PLM community as well as the data analytics community.

Throughout the five years (2014-2018), four journal articles and nine conference papers had been published based on the work leading to this dissertation. Furthermore, a provisional patent had been filed on January 2018.

Patent No.: U.S. Provisional Patent Application No. 62/61385 (SU100944 / 156P574)
Patent title: Smart Products Lifecycle Management Platform for Unmanned Aircraft Systems
Filing date: January 5, 2018
Patent status: Patent Pending

7.3 Research Limitation and Future Work

Product and Process Modeling

While we employed the tensor theory to successfully decompose the smart products hypercube space, we haven't fully investigated how tensor decompositions (CP, Tucker, and other possible decomposition methods) can help achieve optimal modular/integral designs of smart products. Particularly, tensor decompositions have been applied in data mining and machine learning applications recently (Papalexakis et al., 2016; Rabanser et al., 2017); the tensor techniques are also gaining more research interest in temporal network analysis and community detection (Anandkumar et al., 2014; Gauvin et al., 2014). These new approaches would be complementary to the existing DSM/DMM approaches for product and process modeling and analysis in higher dimensions. Furthermore, we had derived a modularity quantification equation for analytics model architecture design, and this equation is similar as that for the architecture of physical products. How to evaluate the modularity of the overall smart product architecture, including physical components and analytics models, would be naturally a future study.

Process-wise, the NPD^3 model developed in this research is a starting point to facilitate understanding how engineers and data scientists should collaborate when they collectively need to develop future smart products that are highly data-driven. As with many empirical studies, however, the generality of our findings could be enhanced by conducting additional case studies for other smart

products in different industries, as suggested by Gibbert and Ruigrok (2010) regarding how to address the rigor of case studies. We also realize the descriptive nature to present the interaction patterns between engineers and data scientists. Hence, as more case studies are conducted, these patterns and characteristics could be further analyzed for quantitative evaluation and comparison. Last but not least, we only focused on the concept development stage in this paper, similar analyses could be conducted on other stages (e.g. *Sub-System Design* and *Detail Design*) of the product development process. As shown in Figure 7.1, we also developed an overall BPMN diagram for the integration of the entire NPD and CRISP-DM processes at the macro-level. The contents and patterns of detailed information flows can be studied using the similar approach that was introduced in Chapter 5.

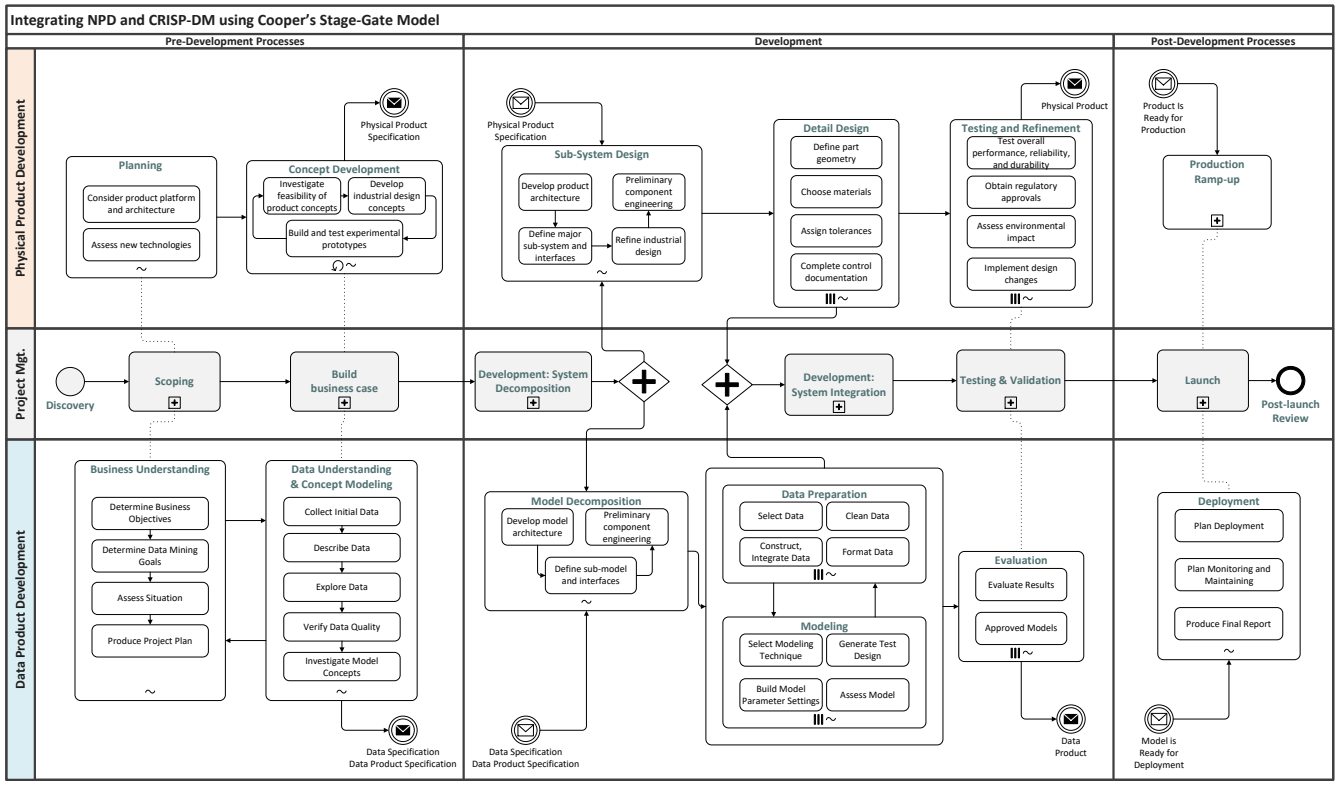


Figure 7.1 Integrate traditional NPD and CRISP-DM at all stages

Implementation of Information Model Standards

When we implemented the data analytics standards in the PLM system, there remained some

important issues which need further attention. It is noted that not all analytics models, specifically defined by newly developed algorithms, are covered by the current PMML specification. However, the approach to incorporate new analytics models would be similar to that used in this research, provided the new models can also be formally represented as an Item-Relationship-Item structure.

More importantly, the efforts to standardize analytics models are continuing. For instance, the standardizations of Gaussian Process Regression (GPR) and Bayesian Network (BN) have been added into the PMML specification recently (Park et al., 2017; Nannapaneni et al., 2018). Interestingly, there is a close relationship among Gaussian process, Bayesian network, and the Kalman filter (Ko and Fox, 2008; Reece and Roberts, 2010), which means a large set of state estimate functions (required by smart products) using Kalman filters or extended Kalman filters can be translated into GPR and BN standard models.

Furthermore, the required information granularity for implementing PMML elements in the PLM system, should be standardized at different levels of abstraction wherever possible, to avoid the unnecessary high costs of data storage and data entry. Another challenge is to determine the best practices of modeling the configuration rules for building composition of models. We treat the configuration rules as a rule-set predictive model in this research; however, this assumption requires validation with broader applications.

On the process perspective, the current CRISP-DM reference model is complicated, and is also incomplete. It cannot be fully implemented in a PLM system without putting substantial efforts. The general authentications and authorizations of each activity as well as conditions to trigger/terminate iterations are not yet clearly defined. Processes for model revision and maintenance are not included in the current CRISP-DM specification. These issues would be addressed in future studies.

Future Applications of sPLM

Adapting the sPLM framework to a specific application, the UAS lifecycle management, has shown the sPLM's usability and extendibility. It sheds a light on potentials to extending the sPLM framework to broader unmanned systems and other types of cyber-physical systems, given these products or systems could be projected into the sPH space. While the UAS lifecycle management application presented in Chapter 6 is a "byproduct" of the sPLM research, the application itself can be further developed to support the emerging UAS commercial applications. A working UAS fleet management architecture is shown in Figure 7.2.

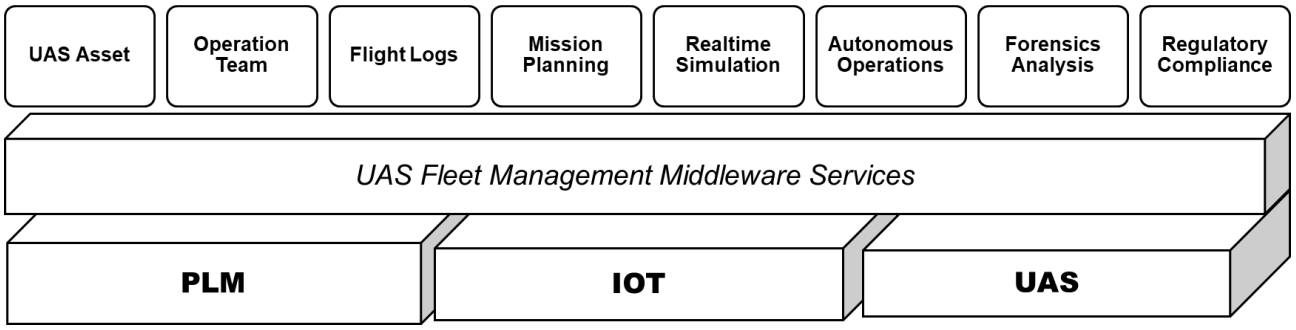


Figure 7.2 PLM-based UAS Fleet Management architecture

Appendix A - Nest Thermostat's Auto-Schedule

The Nest Thermostat consists of simple physical components yet complicated self-learning features that are able to learn users' behaviors and optimize heating and cooling of buildings to conserve energy. Since its debut in 2011, the Nest Thermostat has evolved three generations with ever-increasing smart features. In this section, we present the methods and details for product analysis and process analysis regarding the development of the Nest Thermostat's Auto-Schedule feature.

Table A.1 The Nest Thermostat – three generations

	Generation 1	**Generation 2**		**Generation 3**	
Release	Oct. 27, 2011	Oct. 2, 2012		Sep. 1, 2015	
Size	Diameter 3.2" Depth 1.44"	Diameter 3.27" Depth 1.26"		Diameter 3.3 Depth 1.21	
LCD Display	24-bit color 320 x 320 pixel 1.75" diameter	24-bit color 320 x 320 pixel 1.75" diameter		24-bit color 480 x 480 pixel 2.08" diameter	
Sensors	Temperature (3) Humidity Far-field activity Near-field activity Ambient light	Temperature Humidity Far-field activity Near-field activity Ambient light		Temperature (10) Humidity Far-field activity Near-field activity Ambient light	
Wireless	802.11 b/g/n @2.4 GHz 802.15.4 @2.4 GHz	802.11 b/g/n @2.4 GHz 802.15.4 @2.4 GHz		802.11 b/g/n @2.4 GHz 802.15.4 @2.4 GHz Bluetooth Low Energy (BLE)	
Features	Auto-Schedule Airwave Nest Leaf Auto-Away Energy History Time-to-Temperature	Auto-Schedule Airwave Nest leaf Auto-Away Energy History Time-to-Temperature	System Match Early-On Filter Reminders Heat Pump Balance True Radiant	Auto-Schedule Airwave Nest leaf Auto-Away Energy History Time-to-Temperature	Farsight System Match Early-On Filter Reminders Heat Pump Balance True Radiant Safety Alerts Furnace Heads Up Sunblock Cool to Dry Multi-Zone Home Multi-Home Support
Battery	Rechargeable Lithium-Ion battery	Rechargeable Lithium-Ion battery		Rechargeable Lithium-Ion battery	

In order to understand its physical component architecture, smart features, and the data exchanged

between the local device and the Nest cloud, we employed the Product Archaeology[32] method (Ulrich and Pearson, 1998) to formally investigate and reconstruct the lifecycle of the Nest Thermostat. We purchased a third-generation Nest Thermostat in March 2016 and installed it in our lab. The part list is shown in Table A.2 and the wire connection is shown in Figure A.1.

Table A.2 The Nest Thermostat and the installation accessories

Item	*Specification*
NEST Thermostat	3rd Generation
Elk Transformer with PTC Fuse	24 VAC, 40 VA
Coleman Thermostat Cable	18/3, 50 Feet
Omron General Purpose Relay	LY2-AC24, DPDT
Omron Track/Surface Mount Socket with Finger Protection	For LY1 and LY2 Series Relays

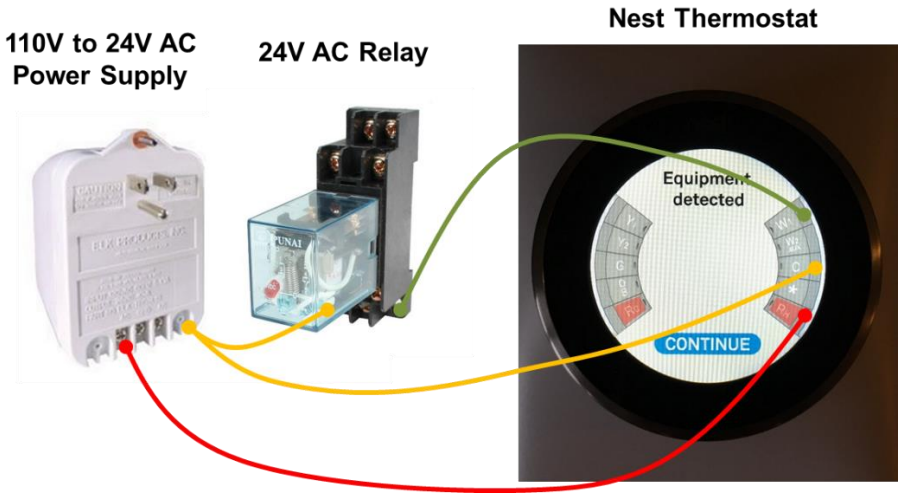


Figure A.1 Nest Thermostat wire connection

Product Analysis

Physically, the Nest Thermostat has two parts: *Main Unit* and *Base Unit*. The main unit contains a round screen and a rotating ring, as well as the main PCB (printed circuit board); the base unit houses the connection terminals. If separated, the display becomes inactive until reconnected to the base. Nest

[32] **Note:** The Product Archaeology approach provides a formal process to reconstruct the lifecycle of the product including its customer requirements, design specifications, and manufacturing processes used to produce it. This approach assumes the data derived from observations of the product itself are objective and highly reliable.

Thermostat is built around Linux and other free software components. It interconnects with other Nest devices using a protocol called Weave[33], which is based on IEEE 802.15.4 and Wi-Fi 802.11 b/g/n.

Physical Components and Analytics Models

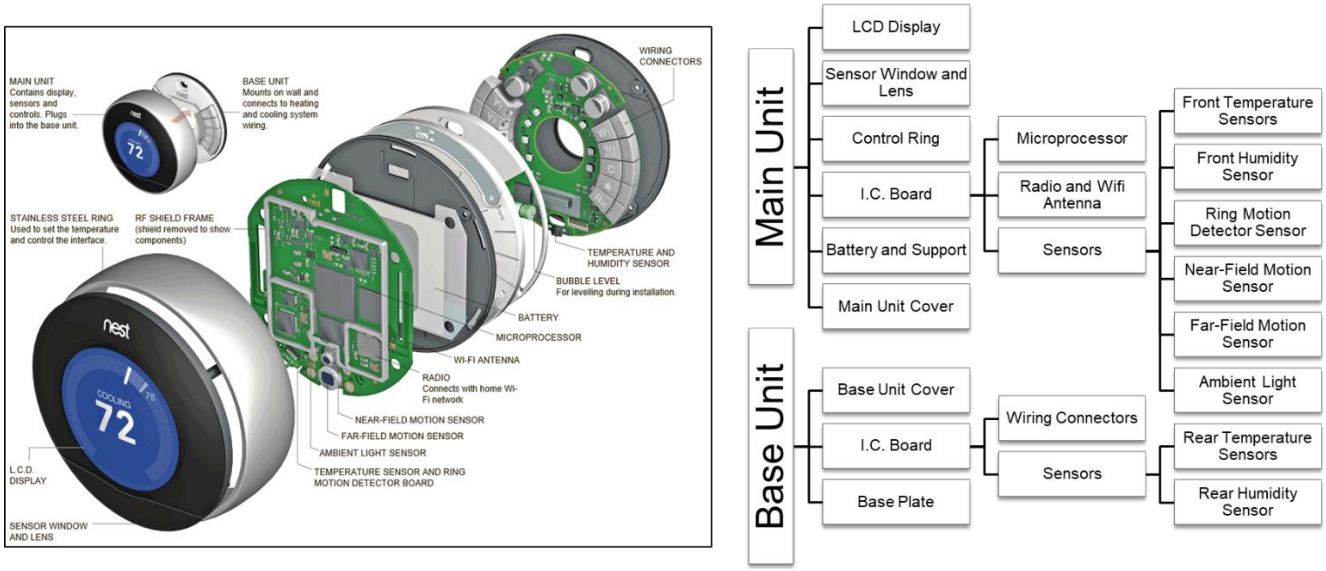


Figure A.2 Nest Thermostat physical components[34] (The dissection image is adapted from NYTimes, 2012)

Table A.3 Nest Thermostat analytics features

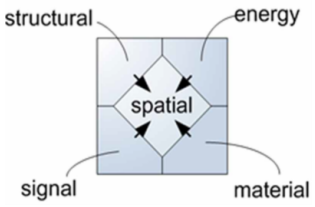
Smart Features	*Description*
Auto-Away	Is based on algorithms that interpret occupancy sensor data and provide a confidence determination of whether or not the occupants are away from the home.
Auto-Schedule	The learning algorithm is based on the user's manual temperature selection on the device, and replays a setback schedule.
Nest Leaf	The Nest Leaf displays when the person controlling the thermostat has chosen an energy-efficient setting.
Time-to-Temperature	A prediction of when the target temperature will be reached.
Early On	A feature to begin heating or cooling the home in advance of an upcoming set point so that the home is at the temperature you want at the time you want it.
True Radiant	A feature that enables better scheduling and avoids overshooting temperatures often associated with radiant heating systems.
Heat Pump Balance	A feature that balances the use of faster-but-more-expensive auxiliary heating with slower-but-less-expensive heat pump heating.
Farsight	A feature that can detect if someone is far away or if they're close and adjust the information displayed accordingly.

[33] Nest Weave Overview, https://developers.nest.com/documentation/weave/weave-overview/
[34] NYTimes, Inside the Nest Learning Thermostat, http://www.nytimes.com/interactive/2012/10/04/business/inside-the-nest-learning-thermostat.html

The Nest Thermostat implements a set of analytics features (*Auto-Away*, *Auto-Schedule*, *Time-to-Temperature*, *Farsight,* etc.) using sophisticated machine learning algorithms. These features are collectively named *Nest Sense*, which is further augmented by the *Nest Cloud* to make it capable of incorporating third-party services like local weather forecast, helping an installed Nest Thermostat learn the user's living pattern and generate an energy-efficient yet comfortable control strategy to the linked HVAC system. We list several analytics features in Table A.3.

DMM for Physical Components and Analytics Models

The correlations between the physical components and analytics features in a Nest Thermostat can be captured and represented in a domain mapping matrix (DMM), which is a rectangular matrix (M by N) to model relationships and interactions across domains (Danilovic and Browning, 2007). According to Sosa et al. (2003), the interfaces among any product components are possibly of five types: Spatial, Structural, Energy, Material, and Information (or Signal). It is obvious that structural interactions often require spatial adjacency of two components, but energy, signal, and material exchanges sometimes also require spatial adjacency. Helmer et al. (2008) proposed a Spatial-Interface-Concentrated approach to handle multiple types of interactions in DSM clustering, which is termed as Perspective Reduction (PR).



2	Both Interaction and Spatial adjacency are required
1	Interaction is required; Spatial adjacency is beneficial
0.5	Both Interaction and Spatial adjacency are beneficial
0	Indifferent
-0.5	Both Interaction and Spatial adjacency are undesired
-1	Interaction is avoided; Spatial adjacency is undesired
-2	Both Interaction and Spatial adjacency are avoided

Figure A.3 Five types of interactions and 7-scale rating schema (adapted from Helmer et al., 2008)

As shown in Figure A.3, this method collapses the four entries (Structural, Energy, Material, and Signal) per cell of a DSM to a single overall dependency mark (Spatial). The idea is based on a modified

7-scale rating schema that allows a comparison of the "importance" of different interaction types (the right part of Figure A.3). For instance, if there is a negative energy exchange between two elements if the elements are close (-1) and a necessary material exchange requires spatial adjacency (+2), it is determined that the requirement for spatial adjacency prevails in order to achieve proper functionality. Hence, the result of this tradeoff is +2 spatial requirement.

Table A.4 The Nest Thermostat's DMM of physical components and analytics models

Physical Components / Analytics Features	LCD display	Sensor window and lens	Control ring	RF shield and frame	Main I.C. Board & Microprocessor	Radio and Wi-fi antenna	Near-field motion sensor	Far-field motion sensor	Ambient light sensor	Front temperature sensor	Front humidity sensor	Battery and support	Main unit cover	Base unit cover	Base I.C. Board & Wiring connectors	Rear temperature sensor	Rear humidity sensor	Base plate
Auto-Away	1	2	0	0	2	1	2	2	0	0	0	0.5	0	0	0	0	0	0
Auto-Schedule	2	0	2	0	2	0.5	1	1	1	2	2	0.5	0	0	2	2	2	0
Time-to-Temperature	2	0	2	0	2	1	0	0	0	2	0	0.5	0	0	2	2	0	0
Early-On	1	0	0	0	2	0	0	0	0	1	0	0.5	0	0	2	1	0	0
Cool to Dry	1	0	0	0	2	0	0	0	0	0	2	0.5	0	0	2	0	2	0
Sunblock	1	2	0	0	2	1	0	0	2	1	0	0.5	0	0	0	0	0	0
Leaf	2	0	2	0	2	0	0	0	0	0	0	0.5	0	0	0	0	0	0
Farsight	2	2	0	0	2	0	2	2	0	0	0	0.5	0	0	0	0	0	0
Nest Cloud	1	0	0	0	1	2	0	0	0	0	0	1	0	0	1	0	0	0
Weather Service	1	0	0	0	0.5	2	0	0	0	0	0	0	0	0	0.5	0	0	0
Energy Service	1	0	0	0	0.5	2	0	0	0	0	0	0	0	0	0.5	0	0	0

We use Helmer's method to calculate the Nest Thermostat's interaction dependencies between physical components and analytics features. For instance, using the LCD screen to display the remaining time to achieve the target temperature is necessary for effective user interaction. The interaction dependency between these two components, the LCD display and the Time-to-Temperature feature, is required. Since the shown temperature is for the building where the user resides, the spatial adjacency between the Time-to-Temperature and the LCD display is also required. Therefore, we mark +2 for this dependency. On the other side, the Main IC Board & Microprocessor does not need to interact directly with the Weather Service, instead, it can receive the weather information through the Nest Cloud. However, it is beneficial if the weather information can be processed locally because this information

contributes to several predictive features. Thus, we mark +0.5 for this dependency. The complete DMM is shown in Table A.4.

Component Clustering: *Community Structure Detection*

Table A.5 Betweenness scores for all dependencies (the edges of the graph)

Physical Components / Analytics Features	LCD display	Sensor window and lens	Control ring	RF shield and frame	Main I.C. Board & Microprocessor	Radio and Wi-fi antenna	Near-field motion sensor	Far-field motion sensor	Ambient light sensor	Front temperature sensor	Front humidity sensor	Battery and support	Main unit cover	Base unit cover	Base I.C. Board & Wiring connectors	Rear temperature sensor	Rear humidity sensor	Base plate
Auto-Away	8.37	8.48	0	0	1.33	2.00	0.87	0.87	0	0	0	22.88	0	0	0	0	0	0
Auto-Schedule	0.77	0	12.61	0	8.67	17.12	22.43	22.43	22.83	1.78	13.27	63.60	0	0	8.83	3.37	13.27	0
Time-to-Temperature	0.27	0	6.28	0	2.67	1.00	0	0	0	0.46	0	20.35	0	0	2.83	0.70	0	0
Early-On	9.58	0	0	0	2.50	0	0	0	0	10.76	0	37.78	0	0	3.00	19.93	0	0
Cool to Dry	11.08	0	0	0	2.00	0	0	0	0	0	10.73	29.55	0	0	2.00	0	10.73	0
Sunblock	9.70	9.33	0	0	1.83	3.00	0	0	1.17	11.01	0	28.13	0	0	0	0	0	0
Leaf	0.27	0	5.12	0	3.67	0	0	0	0	0	0	23.18	0	0	0	0	0	0
Farsight	0.20	6.19	0	0	3.33	0	0.70	0.70	0	0	0	23.26	0	0	0	0	0	0
Nest Cloud	1.00	0	0	0	3.33	0.50	0	0	0	0	0	24.26	0	0	6.43	0	0	0
Weather Service	12.62	0	0	0	7.83	3.17	0	0	0	0	0	0	0	0	6.62	0	0	0
Energy Service	12.62	0	0	0	7.83	3.17	0	0	0	0	0	0	0	0	6.62	0	0	0

The matrix representation of the DMM is then seen as a graph with the matrix's rows and columns as the graph's nodes and the values of cells as the weights of the graph's edges. We apply the community structure detection method proposed by Newman and Girvan (2003). This algorithm employs a divisive method that iteratively removes the edge with the highest "betweenness" score from the graph network to split the network into communities and then recalculates betweenness for all remaining edges and repeat the removal procedure. The betweenness measure is based on the number of shortest (geodesic) paths between pairs of vertices. The betweenness score for each interaction of the Nest Thermostat component network is shown in Table A.5.

The detected communities and their connections are shown in Figure A.4. It is observed that the Auto-Away and Farsight features share the same sensors and they form one community. The Cool to Dry feature is mainly related to humidity sensors and the Sunblock is mainly related to the light sensor. The

Auto-Schedule community and the Nest Cloud community have overlaps. That is because the Time-to-Temperature provides important remaining time prediction to compute the heating/cooling schedule, while the Time-to-Temperature feature needs local weather information that is from the Nest Cloud. The R code for data processing and clustering is shown in Table A.6.

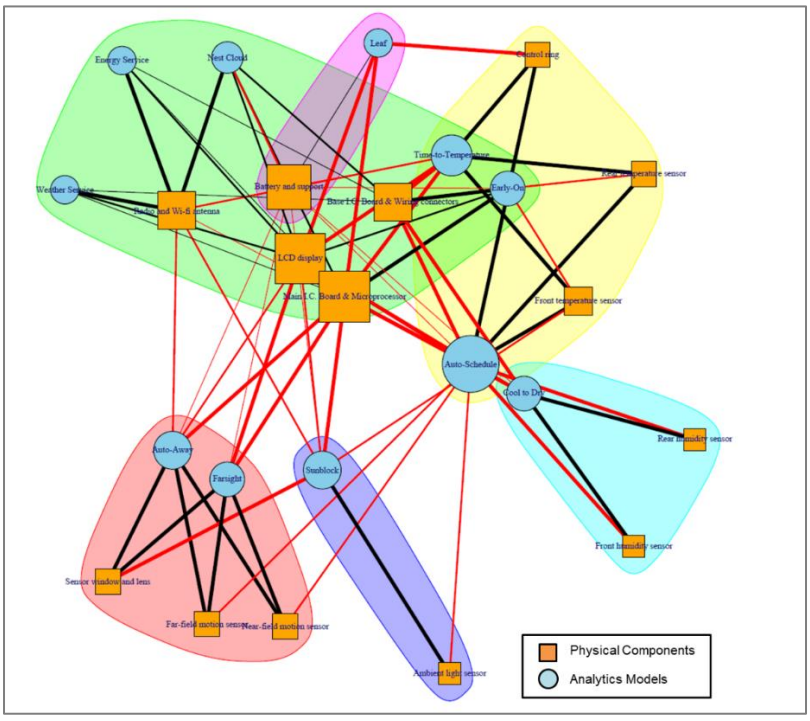


Figure A.4 Clustering of the Nest Thermostat components – Generation 3

Table A.6 R Code for network analysis of the Nest Thermostat's DMM

```
library(rstudioapi)
library(igraph)

# set working directory
current_wd <- getwd()
current_path <- getActiveDocumentContext()$path
setwd(dirname(current_path))

# read adjacency matrix: Nest Thermostat DMM
data_file <- "Nest_Thermostat_Gen3_DMM_P&A.csv"
dat <- read.csv(file=data_file, header=TRUE, row.names=1, check.names=FALSE)
dat[is.na(dat)] <- 0

# remove zero rows and columns
dat.temp <- dat[, colSums(dat)>0]
dat.temp <- dat.temp[rowSums(dat.temp)>0,]
dat <- dat.temp
```

```
# set graph network properties: orange and square shapes are for physical components;
# blue and circle shapes are for analytics features
net <- graph_from_incidence_matrix(dat, weighted=TRUE)
V(net)$color <- c("skyblue", "orange")[V(net)$type+1]
V(net)$shape <- c("circle", "square")[V(net)$type+1]
V(net)$label.color = "midnight blue"
V(net)$label.cex = 0.6
V(net)$label.font = 1 # 1=plain, 2=bold, 3=italic

# construct network layout
lo <- layout_with_fr(net)

# calculate degree and betweenness
deg <- degree(net, mode="all")
v_bet <- betweenness(net, v = V(net), directed=FALSE)
e_bet <- edge_betweenness(net, e = E(net), directed = FALSE)

# communicty detection and plotting
ceb <- cluster_edge_betweenness(net)
par(mfrow=c(1,1), mar=c(0,0,0,0))
plot(ceb, net, col=V(net)$color, vertex.size=5+deg, edge.width=E(net)$weight*2, layout=lo)

# restore working directory
setwd(current_wd)
```

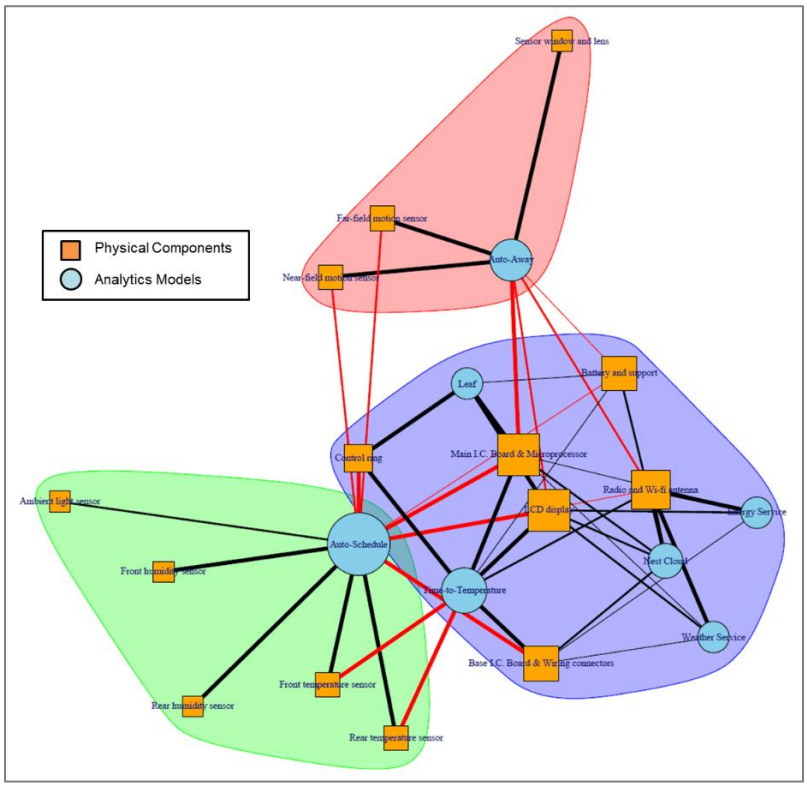


Figure A.5 Clustering of the Nest Thermostat components – Generation 1

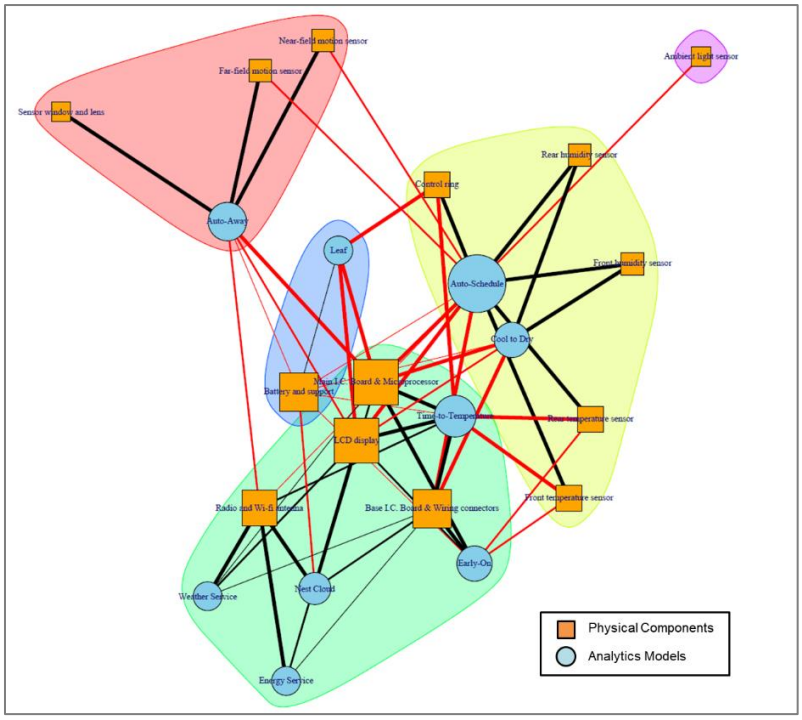


Figure A.6 Clustering of the Nest Thermostat components – Generation 2

Similarly, we can obtain the component DMM and community clusters for the Generation 1 and Generation 2 of Nest Thermostats, see Figure A.5 and A.6.

Process Analysis

Product Development History

A set of technical reports explain the details of the Auto-Schedule feature development and improvement (Nest Labs, 2012-2014). Literatures also reported the study of how users actually perceive and use residential thermostats (e.g., Peffer et al., 2011). The development history of Auto-Schedule is summarized as below, following the steps defined in the New Product Development (NPD) process model.

Customer Needs of a Residential Thermostat

- Set a target temperature, see the current temperature, and control the equipment accordingly
- Save energy but maintain comfort

Feasible Concepts

- Design: mechanical vs. mechatronic
- Shape: round vs. rectangular
- User interaction: physical buttons vs. touch screen vs. auto-sensing
- Schedule programming: non-programmable vs. manual programming vs. auto-learning
- Auto-Schedule: Auto-Schedule without non-occupancy consideration vs. Auto-Away detection

Generic Thermostat Product Architecture

- Layer 1: User interface, Communication interface,
- Layer 2: Sensors, Actuators, Control logic
- Layer 3: Control logic, Memory, Power supply

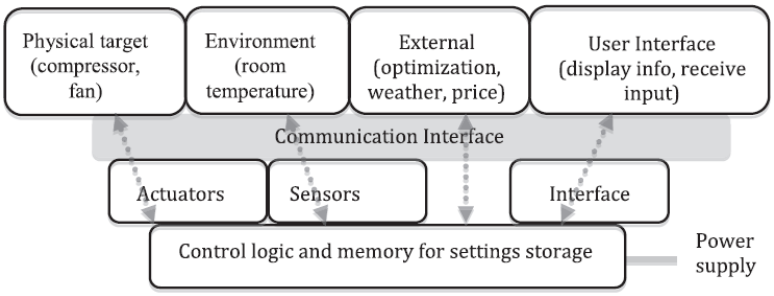


Figure A.7 The generic product architecture of residential thermostat (Peffer et al., 2011)

Nest Marketing Surveys

- Customer need survey: how users adopt intelligent home technologies in their daily life; how users use the device programmed interface; how users teach the device about their preferred cooling and heating settings.
- Customer usage survey: the analysis of data collected in the period between May 2012 and September 2013 from Nest Energy Partner program indicated average saving in southern California of 1.16kWh per day or 11.3% of AC-related energy usage.
- Product performance monitoring: The MyEnergy service allows consumers to monitor energy usage on a monthly and yearly basis and compare the performance with friends' homes or homes in neighborhood. This contributed to insights about energy usage patterns and energy savings achieved by consumer samples.

Auto-Schedule Concept (The First Version, October 2011)

The Nest Learning Thermostat automatically learns a user's preferred temperature as well as schedule. It starts with an empty schedule so that it can learn the routines and preferences of the user, and continues to adapt to their ever-changing schedule.

Simulation Model:

(1) Use Case Scenario

The simulation assumes an 1,800 square feet single-family home with an average efficiency level. Three scenarios are considered: (i) Learned schedule and setpoint without considering any away time; (ii) 1F carving: assume the user change his/her behavior based on Nest Leaf to use 1F less or more as the setpoint; (iii) Auto-away: simulate four weeks of non-occupancy annually.

(2.a) Physics-based Model

The building envelope heat transfer model begins with a standard $U*A*dT$ model, where U is the heat transfer coefficient; A is the surface area of the house, and dT is the difference between the indoor and outdoor temperatures. A number of details are employed in the simulation model to account for important system dynamics that could have

an impact on various thermostat control strategies:

- Air infiltration is based on a detailed infiltration model that includes wind and stack effects using hourly wind speeds, indoor temperature, and outdoor temperature.
- Heating and cooling equipment is modeled to include transient start-up effects, and interactions with thermal mass and distribution systems.
- The heating equipment is assumed to be forced-air gas furnace. For both heating and cooling, single-stage systems are assumed. At the initial stage, heat pump systems, non-forced-air systems, and multi-stage heating and cooling systems are not considered in the model.
- Air conditioner capacity and power draw are modeled as a function of indoor and outdoor conditions with latent (humidity-related) effects of moisture loads from occupants and air infiltration with the dynamic moisture removal capacity of the air conditioner.

(2.b) Analytics-based Model

- Auto-Away: This feature automatically detects non-occupancy events, whether they last for several hours or multiple days. It is based on algorithms that interpret occupancy sensor data and provide a confidence determination of whether or not the occupants are away from the home.
- Auto-Schedule: This feature automatically learns a user's preferred temperature as well as schedule. The learning algorithm is based on the user's manual temperature selection on the device, then the thermostat replays a trained setback schedule.
- The Nest Leaf: A green leaf icon displays when the user has chosen an energy-efficient setting which is determined by the calculated efficiency of the household, the HVAC system model, and the user's prior behavior.
- Time to target temperature: This feature shows the user an exaggerated setpoint temperature takes much longer to reach, which discourages this wasteful behavior.

(3) Weather

A typical year data set of hourly values of solar radiation and meteorological elements is used. The third and latest typical meteorological year (TMY3) data is used, which was developed by the National Renewable Energy Laboratory for the climates simulated. Solar gain through windows is modeled based on hourly solar data from TMY3.

(4) System Parameters

- The heating maintenance band: 1.4F [-1F, +0.4F];
- The cooling maintenance band: 2.0F [-0.7F, +1.3F];
- Heating and cooling temperatures: they are monitored for overshoot. If the room temperature overshoots, the maintenance band is adjusted;
- The minimum cycle time of the air conditioner system: 5 minutes as default;
- The minimum cycle time of the forced-air gas heating system: 3 minutes as default;
- User's window usage: If the room temperature suggests that the HVAC system should turn on, but the outside temperature is at least 5F in the favorable direction, the user is assumed to open the windows and not turn on the HVAC system.

(5) Geographic Locations

A sample of 12 cities (Atlanta, Boston, Chicago, Dallas, Denver, Houston, Miami, Minneapolis, Phoenix, San Diego, San Francisco, and Spokane) in the continental US with at least one city from each of nine US climate zones: Semiarid Steppe climate (Bsk), Humid Subtropical climate (Cfa), Marine Westcoast climate (Cfb), Mediterranean climate (Csa), Humid Continental (warm summer) climate (Dfa), Humid Continental (cool summer) climate (Dfb), Highland (alpine) climate (H), Tropical Wet/Dry Season climate (Aw), Midlatitude Dessert climate (Bwh).

(6) Occupant Type (human factors)

The household occupants are classified into two types: Type 1 (75%) does not have long and regular mid-day non-

occupied periods, e.g., a family with young children, a retired couple, or home office; Type 2 (25%) has predictable, long periods of non-occupancy in the middle of the day, e.g., a working couple having no children, or having children in all-day daycare.

(7) Energy Cost
The cost of the energy varies from city to city. The data from US Energy Information Administration was used.

The overall simulation model is depicted as below:

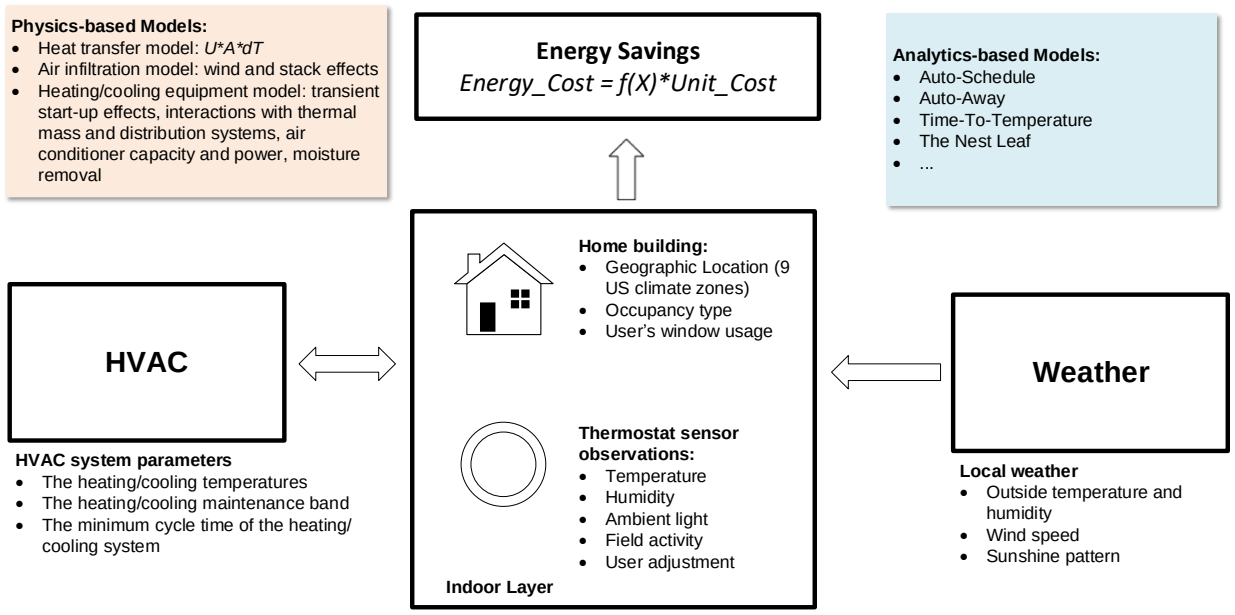


Figure A.8 The Nest Thermostat simulation model

Model Calibration (January-February 2012):
This uses the real field data collected from the first three months of the product release.
Data from 10,000+ installed devices with below criteria for data cleansing:

- Device was not connected to Wi-Fi, or the Wi-Fi connection is less than 90% connection time;
- Device was first installed less than 10 days ago;
- Device did not have a valid US zip code;
- Device controlling heat pump, electric heat, and second stage heat;
- Device in mild climate or device not in heating mode for at least 99% of the period of study;
- Local weather information was missing for more than 3 hours a day.

Enhanced Auto-Schedule (November 2014)
Throughout the year 2012-2014, Auto-Schedule has been steadily updated and improved. The Enhanced Auto-Schedule, released on November 2014, is the first major re-design based on customer feedback and analysis of data from the field across the US, Canada, and the UK.

Goal
The new algorithm should increase efficiency while respecting user inputs and not compromising comfort, and improve responsiveness of schedule learning to changes after the first few weeks.

Updated Simulation Model with Real Data
The simulation data consisted of real, anonymized historical data of user behavior from the first four weeks of device

installation.

Model Verification
A field experiment had been done with two groups of recruited users: half of them used the original algorithm and another half used the enhanced algorithm. The trial was conducted for six weeks during late summer when users were predominantly running air conditioning.
The results showed the Enhanced Auto-Schedule saves users 23.1% in cooling and 19.5% in heating. The setback learned by Enhanced Auto-Schedule is both longer and deeper than it was previously. In addition, the comfort temperature is 1F more efficient than the previously learned schedule.

Decouple Engineering and Data Science Activities

The interactions between engineering and data science activities during concept development can be characterized using the matrix we developed in Chapter 5 (refer to Table 5.6). Below we follow the steps of data understanding (defined in CRISP-DM) to summarize the information content presented in the process interactions. In following tables, PM stands for project management.

Stage 1: *Identify and Collect Data*

This is an **Awareness and Access** interaction process.

Table A.7 Engineering activities versus *Identify and Collect Data*

	Engineering Team (ENG)	PM Team (PJM)	Data Science Team (DST)
Identify and design concepts	▪ Customer technical needs ▪ Competitive technical benchmarking ▪ Historical product technical specification ▪ Product shape, size, aesthetics, ergonomics ▪ User interface ▪ Control and communication interfaces with external environment	▪ Data location and provision: web survey, email survey, literatures, government data, competitive benchmarking, historical product and operation data	▪ Data request: data source, data format, and data time period
	Example. Customers want to achieve greater energy saving while maintaining their comfort in using thermostat. The programmability of existing thermostat products fails to achieve energy saving because users tend not to use them. Historical data is not available to develop the new Auto-Schedule feature of the Nest self-learning thermostat, but competitor products can be referred as baselines to develop the Auto-Schedule feature. On the other hand, historical field data will be available to develop the Enhanced Auto-Schedule feature. Marketing surveys reveal how users can adopt thermostats in their daily life, how users use the device programmed interface, and how users teach thermostats the preferred cooling and heating settings. The dial-type thermostats are more easily adjusted compared to programmable thermostats, providing better		

usability; a bigger LCD screen provides legible feedback information to users. In order to generate an accurate schedule, the thermostat needs to determine when occupants leave the home or go to sleep. Control and communication interfaces with external environment: Wire connection to HVAC systems, sensor locations, and wireless network protocols.

Build and test concepts

Engineering Team (ENG)	PM Team (PJM)	Data Science Team (DST)
▪ Simulation model specification ▪ Product design variables, parameters, constraints, and target performance metrics ▪ What-if scenario descriptions ▪ Simulation data ▪ Filed test data	▪ Physical or virtual sensor data generated from the simulation models or field tests ▪ Public service data ▪ Government data ▪ Customer feedback	▪ Data request: data source, data format, and data time period

Example. For the initial Auto-Schedule development, Nest Thermostat engineers built a simulation model that consists of physics-based models (air infiltration model, heating/cooling equipment model) and analytics-based models (auto-away, auto-schedule, sunblock). This simulation model also takes into consideration of the local weather information (temperature, humidity, wind speed, and sunshine pattern). System variables such as building type, heating/cooling maintenance band, minimum cycle time of the heating/cooling system, resident occupancy, windows usage, and geographic location are included for what-if scenario analyses. Finally, the energy cost saving is used as the performance measure to generate the optimal heating/cooling schedule.

Evaluate concepts for selection

Engineering Team (ENG)	PM Team (PJM)	Data Science Team (DST)
▪ Product bill-of-materials ▪ Baseline performance metrics ▪ What-if simulation and field test data	▪ Historical manufacturing data ▪ Historical use and service/maintenance data ▪ Public and government data	▪ Data request: data source, data format, and data time period

Example. Many Design for 'X' techniques (e.g. Design for Environment) need be conducted to evaluate the product concept so that the manufacturability, serviceability, and sustainability can be achieved to a certain extent while minimizing the total ownership cost. Extensive data is needed to support quantitative decision makings. The product bill-of-materials data is critical to determine the selections of raw materials, manufacturing tools and processes, as well as disassembly and recycling methods.

Stage 2: *Descriptive Analytics*

This is a **Knowledge Transfer** interaction process.

Table A.8 Engineering activities versus *Descriptive Analytics*

Identify and design concepts

Engineering Team (ENG)	PM Team (PJM)	Data Science Team (DST)
▪ Domain knowledge to guide the exploration	▪ Customer needs ▪ Target product specification ▪ Internal and external available data	▪ Conceptual hypotheses ▪ Data visualization and interpretation ▪ Critical features analyses ▪ Further examination suggestion

Example. Based on the assumption that 1-degree carving and auto-away detection can benefit users' energy saving even if they already get used to program their thermostats, Nest proposed the analytics model concepts Auto-Away and the Nest Leaf.

Energy consumption feedback helps the user understand the connection between temperature settings, HVAC use, cost, and the environment. This would enhance the user's perception of how the system is controlled and discourage the use of the thermostat as if it was a valve.

Build and test concepts

Engineering Team (ENG)	PM Team (PJM)	Data Science Team (DST)
▪ Domain knowledge to guide the exploration	▪ Customer needs ▪ Target product specification ▪ Internal and external available data	▪ Conceptual hypotheses ▪ Data visualization and interpretation ▪ Critical features analyses ▪ Further examination suggestion

Example. Every interaction is treated as a way for the user to communicate with the device about his or her preferences for a particular temperature at a particular time any day. Nest also considers the lack of interactions indicates satisfaction with the current temperature.

Evaluate concepts for selection

Engineering Team (ENG)	PM Team (PJM)	Data Science Team (DST)
▪ Domain knowledge to guide the exploration	▪ Customer needs ▪ Target product specification ▪ Internal and external available data	▪ Conceptual hypotheses ▪ Data visualization and interpretation ▪ Critical features analyses ▪ Further examination suggestion

Example. Important features can be identified by exploring the distributions and correlations between design features and the product performance (cost, manufacturability, sustainability)

Stage 3: *Verify Data Quality*

This is a **Knowledge Transfer** interaction process.

Table A.9 Engineering activities versus *Verify Data Quality*

Identify and design concepts

Engineering Team (ENG)	PM Team (PJM)	Data Science Team (DST)
▪ Review data verification report	▪ Data requirement specification ▪ Data qualify assurance	▪ Data verification report ▪ Possible solutions to the unsolved data quality problems

Example. For Nest Thermostat's initial Auto-Schedule development, both historical product data and similar competitive product data may not exist, the absence of such data maybe a data quality problem for concept investigation.

The lack of user behavior (to smart thermostat) data maybe a data quality problem for industrial design concept development.

Build and test concepts

Engineering Team (ENG)	PM Team (PJM)	Data Science Team (DST)
▪ Review data verification report	▪ Data requirement specification ▪ Data qualify assurance	▪ Data verification report ▪ Possible solutions to the unsolved data quality problems

Example. The simulation model may not reflect the real situation, thus the data generated from the simulation model has potential quality problem.

Evaluate concepts for selection

Engineering Team (ENG)	PM Team (PJM)	Data Science Team (DST)
▪ Review data verification report	▪ Data requirement specification ▪ Data qualify assurance	▪ Data verification report ▪ Possible solutions to the unsolved data quality problems

Example. The simulation model may not reflect the real situation, thus the data generated from the simulation model has potential quality problem.

Stage 4: *Investigate Analytics Concepts*

This is a **Problem-Solving** interaction process.

Table A.10 Engineering activities versus *Investigate Analytics Concepts*

Identify and design concepts

Engineering Team (ENG)	PM Team (PJM)	Data Science Team (DST)
▪ Feasible hardware (data storage, processors, sensors, network connections, etc.) and limitations ▪ Sensor (physical or virtual) locations for data generation ▪ Feasible product shape, size, and cost to accommodate sensors and computation	▪ Concept feasibility simultaneously considering hardware, software, analytics, and other supporting techniques ▪ Feasible ergonomic design to facilitate human interactions	▪ Feasible modeling techniques (regression, classification, clustering, association, etc.), and their limitations ▪ Sensor requirement ▪ Repeatable analytics features ▪ Identify feasible and repeatable analytics features ▪ Provide data visualization and interpretation to users

Example. Local weather information is included for Nest Thermostat to determine the air infiltration effect and potential temperature sensing errors due to sunshine on the thermostat. There are several ways to gather the weather information, e.g. using outdoor sensors or through Internet weather service. Data scientist may propose different hypotheses and analytics models to utilize the weather data to predict the impacts of outdoor temperature/humidity changes; while product engineers can propose different mechanisms to obtain, store, even process the weather data.

Human interaction to a thermostat reflects the human feedback to the current room temperature and humidity and intention to change. An ergonomic design not only provides the user error-proof capability, but also guides the user's behavior by providing useful feedback information. The Nest engineers chose a round ring design to allow the user adjusting the temperature back and forth, with showing a leaf icon if the user inputs a target temperate that falls into a specific range. But this temperature range needs to be

determined for each individual user to maintain the comfort. Data scientists can use the target temperature change (and sometimes non-change) data to develop models for user input pattern recognition and behavior prediction.

Build and test concepts

Engineering Team (ENG)	PM Team (PJM)	Data Science Team (DST)
▪ Hardware specification update ▪ Test data update ▪ System variables update	▪ Integrated concepts ▪ Integrated test analyses ▪ Customer feedbacks update	▪ Model performance report ▪ New hypotheses ▪ New analytics model concepts

Example. As more data is available, data scientists can explore more analytics concepts using more advanced big data analytics and machine learning techniques. The performance of the analytics models will change as more data is used. Thus, data scientists may propose new hypotheses and new machine learning algorithms. Real field data from installed instances allow Nest to develop an Enhanced Auto-Schedule feature to accommodate broader use scenarios (US, Canada, UK). More temperature sensors are used to increase the accuracy of temperature prediction.

Evaluate concepts for selection

Engineering Team (ENG)	PM Team (PJM)	Data Science Team (DST)
▪ Cost analysis ▪ Design for Assembly analysis ▪ Design for Manufacturing analysis ▪ Design for Sustainability analysis	▪ Shop floor and production operation data ▪ Manufacturing production constraints ▪ Product component engineering and assembly constraints	▪ Upgradability analysis of analytics features ▪ Improved analytics model concepts to achieve better energy efficiency

Example. The average lifespan of a programmable thermostat would be over ten years. The cost of energy saving might weigh over its initial investment during such a long use term. The upgradability of the thermostat and its adaptability to new HVAC systems are important features to allow the thermostat evolve over its use stage. Throughout the year 2012-2014, the Auto-Schedule feature had been steadily updated and improved. The Enhanced Auto-Schedule feature was released in November 2014 and had been pushed on all three generations of Nest Thermostats in use. In October 2016, Nest announced to upgrade the Auto-Away feature to Eco-Temperature for all three generations of thermostats. This new feature is smarter and can help to save more energy.

Appendix B - Tools Used for sPLM Development and Sample Code

In this section, we explain several tools that are used to implement the sPLM framework (Figure 2.20, Chapter 2) and the UAS lifecycle management platform introduced in Chapter 6.

PLM: *Aras Innovator*

Aras Innovator[35] is an object-oriented, web-based PLM platform as part of a service-oriented architecture (SOA). It provides core PLM functions including product engineering, component engineering, program management, engineering change management, etc. Aras Innovator is highly customizable, using JavaScript as client-side webpage programming and .NET compatible languages (e.g. C#) as server-side webpage and web service development.

ItemType and RelationshipType

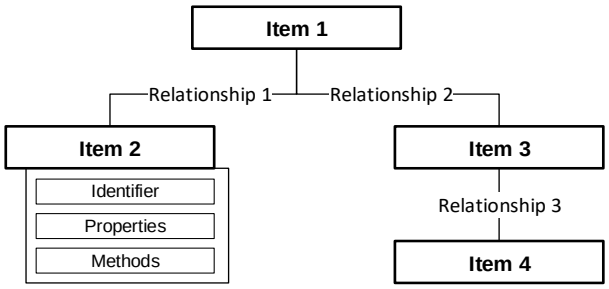


Figure B.1 The Item-Relationship-Item abstraction in Aras Innovator (Li et al., 2015b)

Aras Innovator uses the concepts *Item* and *Relationship* to abstract arbitrary objects and connections between objects. Everything in Aras Innovator is an item, which is an instance of an **ItemType**, which

[35] Aras Innovator, http://www.aras.com/

itself is an item too. Each item has a 32-character GUID (Globally Unique Identifier). An item may have relationships to other items; a relationship is defined by **RelationshipType**, which is also an item. The relationship type rule is defined by using three properties: the source (parent) item, the related (child) item, and the relationship item. This Item-Relationship-Item structure is depicted in Figure B.1.

Lifecycle Map and Workflow Map

Aras Innovator uses two workflow models to support lifecycle activities: state-based and activity-based. The state-based workflow model, which is named **Lifecycle Map**, tracks the state of an item during its lifecycle. A lifecycle map consists of a series of states (actions and steps) and transitions (paths between the different states) that an item instance traverses during its existence. The activity-based model, which is named **Workflow Map**, tracks the work that people actually perform. A workflow map consists of activities and paths, in which each activity represents a unit of work that must be performed. An activity contains the task list, the assignment to users responsible for these tasks, notifications, and time spent on the activity. A workflow map can be accessed from the entry of a lifecycle state and in turn the activities within the workflow map can promote the lifecycle to the next states. The relationships among lifecycle stages and workflow activities are shown in Figure B.2.

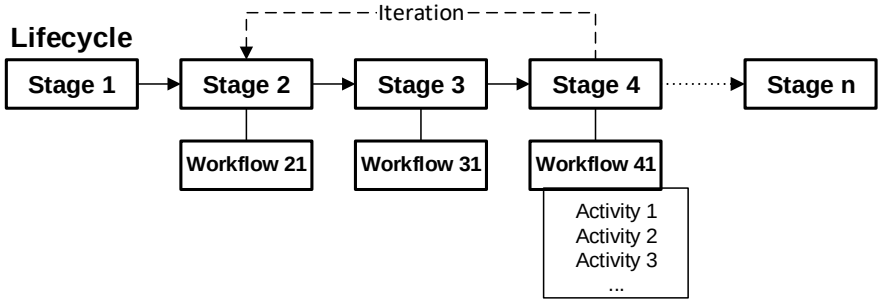


Figure B.2 The lifecycle stage and workflow models in Aras Innovator

Aras Markup Language

The AML (Aras Markup Language) is an XML dialect that follows the Item-Relationship-Item

recursive pattern to describe arbitrary item configurations. Clients submit AML documents to the Aras Innovator server and receive an AML document back. In an AML document, the **<AML>** tag is the root element of an AML document, and may contain one or more **<Item>** tags. Item tags may contain one or more property tags using the property name defined in Aras Innovator as the tag name and a **<Relationships>** tag, which in turn may contain one or more item tags.

In the example shown in Table B.1, the AML query is to retrieve the *PMML* item related to the polynomial regression model expressed in Table 3.2 (Chapter 3). The item tag uses three attributes: *type*, *action* and *id* to define the query condition. The action can be one of the four operators: *get*, *add*, *update*, and *delete*. The Aras Innovator web server returns the response (encoded in AML) once the query is submited. The response includes the details of the requested item, for example, the creation/modification timestamps, the version/revision, the lifecycle state, etc.

Table B.1 An AML examples

```
<AML>
  <Item type="DMG PMML" action="get" id="CCC4DFACDFF84D3A827390CA5F7307CB" />
</AML>

<Result>
  <Item type="DMG PMML" typeId="14F7BC18C09843B3A02E44AA3463B014" id="CCC4DFACDFF84D3A827390CA5F7307CB">
    <item_number>PMML-0015</item_number>
    <created_on>2015-04-22T04:58:16</created_on>
    <modified_on>2015-06-04T22:50:33</modified_on>
    <name>Turning Machining Power Prediction - Aluminum</name>
    <description>Regression for turning machining operation</description>
    <pmml_type>RegressionModel</pmml_type>
    <pmml_application>KNIME 2.9.2</pmml_application>
    <pmml_version>4.1</pmml_version>
    <pmml_file keyed_name="PRL_Turning001_AL_Fri_1.pmml" type="File">
      9DF2269C3726479794E214EB89AC3DA8
    </pmml_file>
    <major_rev>A</major_rev>
    <generation>1</generation>
    <state>Preliminary</state>
  </Item>
</Result>
```

Query

Response

Implementation of Data Analytics Items

In Aras Innovator, we implemented seven model types – *Regression*, *General Regression*, *Gaussian Process*, *Ruleset*, *Trees*, *Neural Network*, and *Scorecard* – defined in the PMML specification (the Gaussian Process was based on the PMML version 4.3). Their structure elements were mapped to

corresponding ItemTypes. We also implemented a Dataset ItemType, which was based on the PMML DataField definition. The enumerations defined in PMML were mapped to Lists in Aras Innovator. Table B.2 lists the ItemTypes, Relationships, and Lists defined for the PMML implementation.

Table B.2 ItemTypes, RelationshipTypes, and Lists to implement the PMML specification

ItemType (31)	*RelationshipType (40)*	*List (26)*
• Dataset	• Dataset_DataField	• PMML Model
• DMG PMML	• Dataset_Visualization	• PMML_ACTIVATION_FUNCTION
• Model Execution	• GaussianProcess_Dictionary	• PMML_CUMULATIVE_LINK_FUNCTION
• PMML Attribute	• GaussianProcess_Kernel	• PMML_FIELD_USAGE_TYPE
• PMML DataField	• GaussianProcess_LT	• PMML_GPR_KERNEL_TYPE
• PMML DerivedField	• GaussianProcess_MiningSchema	• PMML_INVALID_VALUE_TREAT_METHOD
• PMML GaussianProcess Model	• GaussianProcess_Output	• PMML_LINK_FUNCTION
• PMML GaussianProcessFeature	• GPDictionary_GPFeature	• PMML_MINING_FUNCTION
• PMML GaussianProcessTarget	• GPDictionary_GPTarget	• PMML_MISSING_VALUE_TREAT_METHOD
• PMML GeneralRegression Model	• NeuralNetwork_LT	• PMML_Model_Classification
• PMML GPR Dictionary	• NeuralNetwork_MiningSchema	• PMML_NN_NORMALIZATION_METHOD
• PMML GPR Kernel	• NeuralNetwork_Neuron	• PMML_OUTFIELD_RANKORDER
• PMML MiningField	• Neuron_Connection	• PMML_OUTLIER_TREATING_METHOD
• PMML Model	• Neuron_DerivedField	• PMML_OUTPUTFIELD_ALGORITHM
• PMML NeuralNetwork Model	• PMML_DataDictionary	• PMML_OUTPUTFIELD_RANKBASIS
• PMML Neuron	• PMML_ModelElement	• PMML_PREDICATE_OPERATOR
• PMML Neuron Connection	• PMML_PredictiveModel	• PMML_REGRESSIONNORMALIZATION_MTD
• PMML Node	• PMML_TransformationDictionary	• PMML_RESULT_FEATURE
• PMML OutputField	• Predicate_Predicate	• PMML_RULE_FEATURE
• PMML Partition	• Predictive Model BOM	• PMML_RULE_SELECTION_METHOD
• PMML Predicate	• Predictive Model Dataset	• PMML_SCHEMA_DATATYPE
• PMML Predictor	• Predictive Model Document	• PMML_SCHEMA_OPTYPE
• PMML Regression Model	• Regression_LocalTransformation	• PMML_Specification_Version
• PMML RegressionTable	• Regression_MiningSchema	• PMML_TR_MISSING_VALUE_STRATEGY
• PMML Rule	• Regression_RegressionTable	• PMML_TR_NO_TRUE_CHILD_STRATEGY
• PMML RuleSet Model	• RegressionTable_Predictor	• PMML_TR_SPLIT_CHARACTERISTIC
• PMML Score Distribution	• Rule_Predicate	
• PMML Scorecard Model	• RuleSet_LT	
• PMML Characteristic	• RuleSet_MiningSchema	
• PMML Tree Model	• RuleSet_Rule	
• Predictive Model	• Scorecard_LocalTransformation	
	• Scorecard_MiningSchema	
	• Scorecard_Characteristic	
	• SCCharacteristic_Attribute	
	• Tree_LocalTransformation	
	• Tree_MiningSchema	
	• Tree_Node	
	• TreeNode_Partition	
	• TreeNode_ScoreDistribution	
	• TreeNode_TreeNode	

These items had been packed into two Aras Innovator packages: *com.su.innovator.dataanalytics* and *com.su.innovator.dataanalytics.dataset* so that they can be migrated across different installed Aras Innovator instances. We also implemented several methods to represent the behaviors of these items. Below we explain two methods that were implemented using the client-side API and server-side API

provided by Aras Innovator, respectively.

Client-side Method: *Item Structure Visualization*

Since we represent any object as an Item-Relationship-Item structure, it is useful to visualize the structure for an arbitrary item so as to trace its ancestors and descendants. Table B.3 shows an implementation using the Aras Innovator client API (based on JavaScript) to recursively retrieve the children items of a given ItemType. The result is encoded as JSON format. For example, the regression model structure shown in Figure 3.24 (Chapter 3) and the mission structure shown in Figure 6.19 (Chapter 6) were generated by this code.

Table B.3 Client-side method: Retrieve Aras Innovator item structure and return JSON structured data

```
private string GetArasPLMItemStructure(string ItemType, string ItemId) {
    string json = JsonConvert.SerializeObject(GetItemRelationships(ItemType,ItemId));

    return json;
}

private ArasPLMItem GetItemRelationships(string sourceItemType, string sourceItemId) {

  // get the input item name
  Item item = innovator.getItemById(sourceItemType, sourceItemId);
  string itemName = item.getProperty("name");
  if (itemName == null || itemName == "") itemName = item.getProperty("item_name");
  if (itemName == null || itemName == "") itemName = item.getProperty("keyed_name");
  ArasPLMItem sourceNode = new ArasPLMItem();
  sourceNode.name = itemName;

  if(sourceItemType != "File") {
    // get relationships schema
    Item schema_sourceItem = innovator.newItem("ItemType", "get");
    schema_sourceItem.setAttribute("select", "id, name");
    schema_sourceItem.setProperty("name", sourceItemType);
    Item schema_relationshipType = innovator.newItem("RelationshipType", "get");
    schema_relationshipType.setAttribute("select", "related_id(id), name, label, sort_order,
source_id, related_id");
    schema_relationshipType.setAttribute("order_by", "sort_order");
    schema_sourceItem.addRelationship(schema_relationshipType);
    Item schema_query = schema_sourceItem.apply();

    Item schema_relationshipTypeList =
schema_query.getItemByIndex(0).getRelationships("RelationshipType");
    for (int idx = 0; idx < schema_relationshipTypeList.getItemCount(); idx++) {
      schema_relationshipType = schema_relationshipTypeList.getItemByIndex(idx);
      string relationshipTypeName = schema_relationshipType.getProperty("name");
      string relationshipTypeLabel = schema_relationshipType.getProperty("label");
      string related_id = schema_relationshipType.getProperty("related_id");
      if (related_id == null) related_id = "";
      ArasPLMItem relationshipNode = new ArasPLMItem();
      relationshipNode.name = relationshipTypeLabel;

      // get related items
      Item sourceItem = innovator.newItem(sourceItemType, "get");
      sourceItem.setAttribute("select", "id, name, keyed_name");
      sourceItem.setID(sourceItemId);
```

```
        Item relationshipItem = innovator.newItem(relationshipTypeName, "get");
        relationshipItem.setAttribute("select", "related_id(id, keyed_name), name, keyed_name");
        sourceItem.addRelationship(relationshipItem);
        Item item_query = sourceItem.apply().getItemByIndex(0).getRelationships(relationshipTypeName);
        for (int i = 0; i < item_query.getItemCount(); i++) {
          relationshipItem = item_query.getItemByIndex(i);
          string relatedItemName = "";
          string relatedItemType = "";
          string relatedItemId = "";

          if (related_id != "") {
            Item relatedItem = relationshipItem.getRelatedItem();
            relatedItemName = relatedItem.getProperty("keyed_name");
            relatedItemType = relatedItem.getType();
            relatedItemId = relatedItem.getID();
          }
          else {
            Item relatedItem = relationshipItem;
            relatedItemName = relatedItem.getProperty("keyed_name");
            relatedItemType = relatedItem.getType();
            relatedItemId = relatedItem.getID();
          }

          ArasPLMItem relatedNode = GetItemRelationships(relatedItemType, relatedItemId);
          relationshipNode.children.Add(relatedNode);

        }

        if (item_query.getItemCount() > 0) sourceNode.children.Add(relationshipNode);
      }
    }
    return sourceNode;
  }
```

Server-side Method: *Predictive Model Scoring*

The predictive model scoring process (introduced in Section 3.3.5, the analytics model retrieval and execution) can either use the dataset and model raw files stored in the PLM vault or use the metadata for the dataset and model stored in the PLM database. Table B.4 shows a server-side method to locate the dataset and model files related to a predictive model. The method then invokes the JPMML[36] scoring library to execute the scoring procedure.

Table B.4 Server-Side method: Using JPMML for predictive model scoring

```
// retrieve root item
Innovator innovator = this.getInnovator();
string resultMsg = "";

Item paModel = this.newItem(this.getType(), "get");
paModel.setID(this.getID());
Item models = innovator.newItem("Predictive Model Document", "get");
models.setAttribute("select", "related_id(item_number, name, pmml_file)");
Item datasets = innovator.newItem("Predictive Model Dataset", "get");
datasets.setAttribute("select", "related_id(item_number, name, data_file)");
```

[36] JAVA PMML API, https://github.com/jpmml

```
paModel.addRelationship(models);
paModel.addRelationship(datasets);

Item queryResults = paModel.apply();
paModel = queryResults.getItemByIndex(0);

// retrieve model item
Item modelList = paModel.getRelationships("Predictive Model Document");
Item modelRel = modelList.getItemByIndex(0);
Item model = modelRel.getRelatedItem();
string modelFileID =  model.getProperty("pmml_file") ;

// retrieve dataset item
Item datasetList = paModel.getRelationships("Predictive Model Dataset");
Item datasetRel = datasetList.getItemByIndex(0);
Item dataset = datasetRel.getRelatedItem();
string datasetFileID = dataset.getProperty("data_file");

// retrieve model file and dataset file
string workDir = "D:/Aras/Workspace/";
Item modelFile = innovator.newItem("File", "get");
modelFile.setAttribute("select", "filename");
modelFile.setID(modelFileID);
Item queryModelFile = modelFile.apply();
modelFile = queryModelFile.getItemByIndex(0);
string modelFilePath = workDir + modelFile.getProperty("filename");
model.fetchFileProperty("pmml_file", modelFilePath, FetchFileMode.Normal);

Item datasetFile = innovator.newItem("File", "get");
datasetFile.setAttribute("select", "filename");
datasetFile.setID(datasetFileID);
Item queryDatasetFile = datasetFile.apply();
datasetFile = queryDatasetFile.getItemByIndex(0);
string datasetFilePath = workDir + datasetFile.getProperty("filename");
dataset.fetchFileProperty("data_file", datasetFilePath, FetchFileMode.Normal);

// execute scoring and encode the result in HTML format
pmml_scoring_engine_dll.PMMLScoringEngine se = new pmml_scoring_engine_dll.PMMLScoringEngine();
se.inputFile = "\"" + datasetFilePath + "\"";
se.modelFile = "\"" + modelFilePath + "\"";
se.outputFile = "\"" + workDir + "output_" + datasetFile.getProperty("filename") + "\"";
resultMsg += "<h2>Scoring Results:</h2>";
resultMsg += "<b>Model:</b> " + model.getProperty("item_number") + ", " + model.getProperty("name") +
"<br>";
resultMsg += "<b>Dataset:</b> " + dataset.getProperty("item_number") + ", " +
dataset.getProperty("name")+"<br><br>";
resultMsg += se.PMMLScoring();

// return the scoring result to the client
return innovator.newResult(resultMsg);
```

Implementation of sPLM for UAS Items

The UAS data schema illustrated in Figure 6.7 (Chapter 6) was mapped to Aras Innovator by adding the ItemTypes and RelationshipTypes shown in Table B.5. Some regulatory rules were implemented as methods for individual items. For instance, Table B.6 shows the server-side method to check the UAV payload weight.

Table B.5 ItemTypes, RelationshipTypes, and Lists to implement the UAS lifecycle management modules

ItemType (10)	*RelationshipType (18)*	*List (8)*
• UAS Autopilot • UAS Environment • UAS Flight Log • UAS Mission • UAS Mission Planner • UAS Payload • UAS Regulation • UAS UAV • UAS Water Quality Data • UAS Waypoint	• UAS_Autopilot_Parameter • UAS_Environment_Boudary • UAS_Envrionment_Map • UAS_FL_Datasets • UAS_FL_RawLogs • UAS_Mission_Autonomy • UAS_Mission_Environment • UAS_Mission_FlightLog • UAS_Mission_UAV • UAS_Mission_Waypoints • UAS_Regulation_Summary • UAS_UAV_Avionics • UAS_UAV_DigitalModel • UAS_UAV_Payloads • UAS_UAV_PowerSystem • UAS_UAV_Specification • UAS_WP_Regions • UAS_WP_Waypoints	• UAS_MAV_AUTOPILOT • UAS_MAV_CMD • UAS_MAV_FRAME • UAS_MAV_TYPE • UAS_Payload_Category • UAS_Payload_Type • UAS_UAV_Category • UAS_UAV_Class

Table B.6 Server-Side method: UAV payload weight check

```
Innovator innovator = this.getInnovator();
string resultMsg = "";

if(String.Equals(this.getType(), "UAS UAV", StringComparison.Ordinal))
{
  string payload = this.getProperty("payload_gram");
}

/**
 * the logic to evaluate the payload weight, for example, the weight should be less than 55 lbs.
 */

resultMsg = "<font color=red>Warning: There is violation of the UAV payload limit.</font>" +
"<br/><br/>Current Item Type: <b>" + this.getType() + "</b>";

return innovator.newError(resultMsg);
```

These items had been packed into the Aras Innovator package *com.su.innovator.uas* so that they can be migrated across Aras Innovator instances.

Data Analytics Platform: *KNIME*

KNIME Analytics Platform[37] (for short, KNIME) is an integrated, workflow-driven data analytics

[37] KNIME, https://www.knime.org/

platform that allows performing sophisticated statistics and data mining tasks. Its visual workbench combines data access, data transformation, initial investigation, predictive analytics and visualization. KNIME also integrates R[38], Python[39], and JavaScript engines to allow programming with these popular script languages directly within the KNIME modeling environment. Almost all basic KNIME nodes that create an analytics model can represent the model in PMML (if the PMML standard supports it).

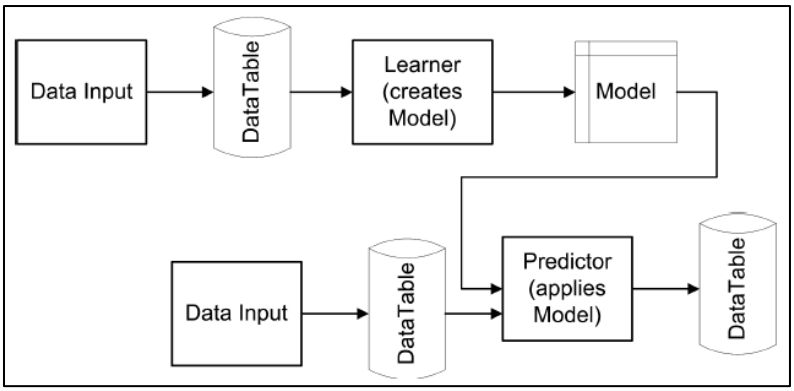


Figure B.3 The semantic flow of data and models in a KNIME workflow (Berthold et al., 2009)

Below we briefly describe how we implemented the Aras Innovator node for KNIME that was introduced in Section 3.3.6 (Chapter 3). The procedure to create a node template (using Java language) can be referred to the KNIME developer guide[40]. KNIME uses an Eclipse plugin to generate necessary Java classes and registers the node into the KNIME repository. Table B.7 shows the Java code for implementing the node user interface shown in Figure 3.25 (Chapter 3) and Table B.8 shows a code snippet to create the node model to allow KNIME to communicate with the Aras Innovator server.

For the *LoginAras* and *ApplyAML* functions used in Table B.8, Table B.9 shows their code, which is adapted from CSDN[41].

---

[38] The R Project for Statistical Computing, https://www.r-project.org/
[39] Python, https://www.python.org/
[40] KNIME: New Node Wizard, https://www.knime.com/developer/documentation/wizard

[41] Java Access Aras Innovator through XML and SOAP, https://blog.csdn.net/plm888/article/details/10314303

Table B.7 The ArasInnovator node for KNIME: the node dialog (user interface)

```
protected ArasInnovatorNodeDialog() {
    super();

    // Connection setting
    createNewGroup("Aras Innovator Connection");
    addDialogComponent(new DialogComponentString(
        new SettingsModelString(ArasInnovatorNodeModel.CFGKEY_SERVERURL,
ArasInnovatorNodeModel.DEFAULT_SERVERURL),
        "Server URL:", true, 35));

    // Aras Innovator login: username
    addDialogComponent(new DialogComponentString(
        new SettingsModelString(ArasInnovatorNodeModel.CFGKEY_USERNAME,
ArasInnovatorNodeModel.DEFAULT_USERNAME),
        "Username:", true, 10));

    // Aras Innovator login: password
    addDialogComponent(new DialogComponentPasswordField(
        new SettingsModelString(ArasInnovatorNodeModel.CFGKEY_PASSWORD, null),
        "Password:", 10));

    // Aras Innovator database list
    ArrayList<String> db = new ArrayList<String>();
    db.add("InnovatorSolutions11");
    db.add("InnovatorSample11");
    addDialogComponent(new DialogComponentStringSelection(
        new SettingsModelString(ArasInnovatorNodeModel.CFGKEY_DBNAME,
ArasInnovatorNodeModel.DEFAULT_DBNAME),
        "Select database:", db, true));
    closeCurrentGroup();

    // AML query entry
    createNewGroup("AML Query");
    addDialogComponent(new DialogComponentMultiLineString(
        new SettingsModelString(ArasInnovatorNodeModel.CFGKEY_AML,
ArasInnovatorNodeModel.DEFAULT_AML),
        "AML String:", false, 35, 6));
    closeCurrentGroup();

}
```

Table B.8 The ArasInnovator node for KNIME: the node model logic

```
public class ArasInnovatorNodeModel extends NodeModel {

  [...]

  /** the settings key which is used to retrieve and
      store the settings (from the dialog or from a settings file)
     (package visibility to be usable from the dialog). */
  static final String CFGKEY_SERVERURL = "ServerURL";
  static final String CFGKEY_USERNAME = "Username";
  static final String CFGKEY_PASSWORD = "Password";
  static final String CFGKEY_DBNAME = "DBName";
  static final String CFGKEY_AML = "AML";

  /** initial default count value. */
  static final int DEFAULT_COUNT = 100;
  static final String DEFAULT_SERVERURL = "http://localhost/InnovatorServer/ ";
  static final String DEFAULT_USERNAME = "admin";
  static final String DEFAULT_PASSWORD = "";
  static final String DEFAULT_DBNAME = "InnovatorSample11";
  static final String DEFAULT_AML = "<AML>\n  <Item type=\"DMG PMML\" action=\"get\" \n
select=\"item_number, pmml_name, pmml_description\" \n   orderBy=\"item_number\">\n  </Item>\n</AML>";

  // example value: the models count variable filled from the dialog
```

```
  // and used in the model's execution method. The default components of the
  // dialog work with "SettingsModels".
  private final SettingsModelString m_serverurl =
      new SettingsModelString(ArasInnovatorNodeModel.CFGKEY_SERVERURL,
ArasInnovatorNodeModel.DEFAULT_SERVERURL);
  private final SettingsModelString m_username =
      new SettingsModelString(ArasInnovatorNodeModel.CFGKEY_USERNAME,
ArasInnovatorNodeModel.DEFAULT_USERNAME);
  private final SettingsModelString m_password =
      new SettingsModelString(ArasInnovatorNodeModel.CFGKEY_PASSWORD,
ArasInnovatorNodeModel.DEFAULT_PASSWORD);
  private final SettingsModelString m_dbname =
      new SettingsModelString(ArasInnovatorNodeModel.CFGKEY_DBNAME,
ArasInnovatorNodeModel.DEFAULT_DBNAME);
  private final SettingsModelString m_aml =
      new SettingsModelString(ArasInnovatorNodeModel.CFGKEY_AML, ArasInnovatorNodeModel.DEFAULT_AML);


  /**
   * Constructor for the node model.
   */
  protected ArasInnovatorNodeModel() {

      // the node has zero incoming port and one outgoing port
      super(0, 1);

  }

  [...]

  /**
   * {@inheritDoc}
   */
  @Override
  protected BufferedDataTable[] execute(final BufferedDataTable[] inData,
          final ExecutionContext exec) throws Exception {

    // Login Aras
    String UserName = m_username.getStringValue();
    String Password = m_password.getStringValue();
    String ServerURL = m_serverurl.getStringValue();
    String DBName = m_dbname.getStringValue();
    Boolean LoginStatus = loginAras(UserName, Password, ServerURL, DBName);

    // Apply AML
    String AMLQuery = m_aml.getStringValue();
    String AMLResponse = ApplyAML(UserName, Password, ServerURL, DBName, AMLQuery, "ApplyAML");


    // the data table spec of the single output table,
    // the table will have three columns:
    DataColumnSpec[] allColSpecs = new DataColumnSpec[5];
    allColSpecs[0] = new DataColumnSpecCreator("Database", StringCell.TYPE).createSpec();
    allColSpecs[1] = new DataColumnSpecCreator("Login User", StringCell.TYPE).createSpec();
    allColSpecs[2] = new DataColumnSpecCreator("Login Status", StringCell.TYPE).createSpec();
    allColSpecs[3] = new DataColumnSpecCreator("AML Query", StringCell.TYPE).createSpec();
    allColSpecs[4] = new DataColumnSpecCreator("AML Response", StringCell.TYPE).createSpec();

    DataTableSpec outputSpec = new DataTableSpec(allColSpecs);

    // the execution context will provide us with storage capacity, in this
    // case a data container to which we will add rows sequentially
    // Note, this container can also handle arbitrary big data tables, it
    // will buffer to disc if necessary.
    BufferedDataContainer container = exec.createDataContainer(outputSpec);
    // let's add m_count rows to it
    RowKey key = new RowKey("Row 0");
    // the cells of the current row, the types of the cells must match
    // the column spec (see above)
    DataCell[] cells = new DataCell[5];
```

```
        cells[0] = new StringCell(DBName);
        cells[1] = new StringCell(UserName);
        cells[2] = new StringCell(LoginStatus.toString());
        cells[3] = new StringCell(AMLQuery);
        cells[4] = new StringCell(AMLResponse);
        DataRow row = new DefaultRow(key, cells);
        container.addRowToTable(row);

        // check if the execution monitor was canceled
        exec.checkCanceled();
        exec.setProgress(1, "Adding row 0");

        // once we are done, we close the container and return its table
        container.close();
        BufferedDataTable out = container.getTable();
        return new BufferedDataTable[]{out};
    }

}
```

Table B.9 Connect to the Aras Innovator server and send AML request to the server

```
// connect to Aras Innovator server
protected static boolean loginAras(String UserName,String Password,String ServerURL,String DBName)
    throws IOException {
    StringBuffer sb = new StringBuffer();
    URL urlconn = new URL(ServerURL);
    HttpURLConnection httpconn = (HttpURLConnection)urlconn.openConnection();
    httpconn.setConnectTimeout(3000);
    httpconn.setRequestProperty("Content-Type", "text/xml; charset=utf-8;");
    httpconn.setDoInput(true);
    httpconn.setDoOutput(true);

    Password = MD5calc(Password);
    httpconn.setRequestProperty("SOAPAction", "ValidateUser");
    httpconn.setRequestProperty("AUTHUSER", UserName);
    httpconn.setRequestProperty("AUTHPASSWORD", Password);
    httpconn.setRequestProperty("DATABASE", DBName);
    String amlstring="<?xml version='1.0' encoding='utf-8'?><Item/>";
    OutputStream outSWrite = httpconn.getOutputStream();
    outSWrite.write((amlstring).getBytes());

    outSWrite.flush();
    outSWrite.close();
    InputStreamReader inSRead = new InputStreamReader(httpconn.getInputStream());

    if(inSRead.ready())
    {
        while(true)
        {
            byte temp=(byte)inSRead.read();
            if(temp==(-1))
            {
                break;
            }
            char tempChar=(char)temp;
            sb.append(tempChar);
        }
    }


    if(sb.toString().contains("<SOAP-ENV:Envelope xmlns:SOAP-
ENV=\"http://schemas.xmlsoap.org/soap/envelope/\"><SOAP-ENV:Body><Result>"))
    {
        System.out.print("\nAras Innovator: Login Successfully.");
        inSRead.close();
        return true;
    }
```

```
    else
    {
        System.out.print("\nAras Innovator: Login Failed.");
        inSRead.close();
        return false;
    }

}

// decrypt the MD5-encoded password
protected static String MD5calc(String md5) {
     […]
}

// send AML query to Aras Innovator server
public static String ApplyAML(String UserName, String Password, String ServerURL,
        String DBName,String AMLString,String soapAction) {
    try {
        StringBuffer sb = new StringBuffer();
        URL urlconn = new URL(ServerURL);
        HttpURLConnection httpconn = (HttpURLConnection) urlconn
                .openConnection();
        httpconn.setRequestMethod("POST");
        httpconn.setRequestProperty("Content-Type", "text/xml; charset=utf-8;");
        httpconn.setDoInput(true);
        httpconn.setDoOutput(true);
        Password = MD5calc(Password);
        if(soapAction!=null)
        {//ApplyItem
          httpconn.setRequestProperty("SOAPAction", soapAction);
        }
        else
        {
            httpconn.setRequestProperty("SOAPAction", "");
        }
        httpconn.setRequestProperty("AUTHUSER", UserName);
        httpconn.setRequestProperty("AUTHPASSWORD", Password);
        httpconn.setRequestProperty("DATABASE", DBName);

        OutputStream outSWrite = httpconn.getOutputStream();
        outSWrite.write((AMLString).getBytes());
        outSWrite.flush();
        outSWrite.close();

        InputStream inSRead = httpconn.getInputStream();
        BufferedReader br=new BufferedReader(new InputStreamReader(inSRead));
        while(true)
        {
            String data=br.readLine();
            if(data==null) break;
            sb.append(data);

        }
        return sb.toString();
    }
    catch(Exception ex)
    {
        ex.printStackTrace();
        return "AML Syntax Error";
    }

}
```

3D Virtual Earth: *CesiumJS and CZML*

The Cesium Markup Language (CZML)[42] was developed by Analytical Graphics, Inc. (AGI)[43]. CZML is designed for Earth scientists to describe and analyze large-scale and time-dynamic objects on 3D virtual globes. It is a JSON-based schema to represent the geometry, appearance, and properties of geospatial objects, and accurately specifies how they change with time. Furthermore, CZML is designed as packets so the data encoded in CZML can be transferred in streaming mode, making it ideal for real-time, data-driven visualization and analysis in web-based applications.

```
[
 // Packet 1: The first packet in CZML is a document object, serving as the document declaration.
 {"id":"document", // The id property of the first packet must be "document".
  "version":"1.0" // The CZML version being written. Only valid on the document object.
 },

 // Packet 2: The second packet.
 {"id":"myObject1", //The ID of the object described by this packet. Any unique identifier.
  "Property 1": property value 1,
  "Property 2":{"Sub-property 1": sub-property 1,
                "Sub-property 2": sub-property 2,
                 // ... Other sub-properties.
                },
  // ... Other properties.
 },

 // Packet 3: The third packet.
 {"id":"myObject2",
  "Property 1": property value 1,
  "Property 2": property value 2
 },

 // Packet 4
 {"id":"myObject2",
  "Property 3": property value 3,
  // ... Other properties.
 },

 // ... Other packets.
]
```

Figure B.4 CZML general structure (Zhu et al., 2018a)

CZML supports static and dynamic properties for many objects defined in its specification. The value of a static property is constant across the entire lifecycle of the described object. Examples of

[42] CZML Guide, https://github.com/AnalyticalGraphicsInc/czml-writer/wiki/CZML-Guide
[43] Analytical Graphics, Inc. https://www.agi.com/

static properties include *id*, *name*, *parent*, *availability*, and *version*. The value of a dynamic property can change over time intervals or epochs. Furthermore, some CZML properties support interpolating an unknown property value at any given time based on given discretized time-based values. For instance, the position property of an object, the color and scale sub-properties of the geometry property of an object, etc. A detailed review of CZML and its applications can be seen in Zhu et al. (2018a; 2018b).

Appendix C - ICUAS Publications by Topics

Table C.1 ICUAS[44] publications distribution (2013-2018)

Keyword	*2013*	*2014*	*2015*	*2016*	*2017*	*2018*	*Total*	*Mean*	*Standard Deviation*
Air Vehicle Operations	11	5	12	11	9	13	61	10	2.61
Airspace Control	19	13	14	13	16	8	83	14	3.34
Airspace Management	5	3	3	5	5	6	27	5	1.12
Airworthiness	1	2	4	3	2	1	13	2	1.07
Autonomy	28	29	37	28	57	49	228	38	11.28
Biologically Inspired UAS	1	6	3	2	5	5	22	4	1.80
Certification	1	3	3	2	0	1	10	2	1.11
Control Architecture	35	41	35	39	58	43	251	42	7.80
Energy Efficient UAS	5	3	8	8	14	4	42	7	3.65
Environmental Issues	3	6	5	5	8	8	35	6	1.77
Fail-Safe Systems	7	5	6	11	11	8	48	8	2.31
Frequency Management	0	0	1	1	1	1	4	1	0.47
Integration	10	10	10	10	13	9	62	10	1.25
Interoperability	1	3	0	2	1	1	8	1	0.94
Levels of Safety	3	1	4	5	7	5	25	4	1.86
Manned/Unmanned Aviation	17	15	20	15	14	12	93	16	2.50
Micro- and Mini- UAS	35	33	37	34	41	24	204	34	5.16
Navigation	31	23	34	38	49	21	196	33	9.39
Networked Swarms	9	10	6	5	12	8	50	8	2.36
Path Planning	23	21	30	43	45	35	197	33	9.14
Payloads	5	6	8	8	16	12	55	9	3.76
Regulations	2	7	7	5	3	2	26	4	2.13
Reliability of UAS	8	8	5	7	11	10	49	8	1.95
Risk Analysis	2	3	2	7	7	6	27	5	2.22
Security	0	4	4	9	8	2	27	5	3.15
See-and-avoid Systems	5	10	13	14	19	9	70	12	4.38
Sensor Fusion	19	12	26	20	26	20	123	21	4.75
Simulation	20	34	33	33	42	23	185	31	7.34
Smart Sensors	5	7	8	3	7	7	37	6	1.67
Standardization	0	3	1	1	1	1	7	1	0.90
Swarms	9	14	11	11	16	14	75	13	2.36
Technology Challenges	7	17	15	14	12	17	82	14	3.45
Training	1	2	3	0	1	2	9	2	0.96
UAS Applications	46	56	68	62	93	56	381	64	14.78
UAS Communications	2	11	6	6	8	10	43	7	2.97
UAS Testbeds	0	0	9	11	13	16	49	8	6.15

[44] The International Conference on Unmanned Aircraft Systems (ICUAS), http://www.uasconferences.com/

Appendix D - Small UAS Remote Pilots Survey

I. Demographics

Gender: ☐ Male ☐ Female

State/Country: _____

Q1.1: How many flights/operations are you conducting each month?

☐ 0-5 ☐ 6-10 ☐ 11-20 ☐ 21-30 ☐ 30+

Q1.2: What is the average duration of each flight/operation?

☐ 0-3 minutes ☐ 4-10 minutes ☐ 11-20 minutes
☐ 21-40 minutes ☐41-60 minutes ☐ Greater than 60 minutes

Q1.3: What is your current investment for your UAS platform(s), including hardware and software?

☐ Less than $500 ☐ $501-$1500 ☐ $1501-$2500
☐$2501-$5000 ☐ Greater than $5000

Q1.4: What is your current liability insurance investment for your UAS platform(s) per year?

☐ Less than $500 ☐ $501-$1000 ☐ $1001-$2000
☐$2001-$5000 ☐ Greater than $5000

UAS Operations

Q2.1: In what capacity do you primarily operate a UAS or "drone"? (select all that apply)
☐ Recreational
☐ Independent contractor
☐ Contract/Part-time/Full-time employee
☐ UAS business owner
☐ I don't operate a UAS

Q2.2: If you or your organization are operating a UAS fleet, what is the size of the UAS fleet?
☐ Less than 5 ☐ 5-10 ☐ 10-25 ☐ Greater than 25

Q2.3: In what areas are you using a UAS? (select all that apply)
☐ Recreation/Hobbyist
☐ Agriculture
☐ Cinematography
☐ Construction
☐ Defense/Security
☐ Energy/Power/Utilities
☐ Healthcare
☐ Industrial (Manufacturing, warehouses, seaports)
☐ Inspections

☐ Insurance
☐ Mapping/Surveying
☐ Mining
☐ Natural Resource Management
☐ Oil/Gas
☐ Photography (Professional, Media, Publications)
☐ Public Safety
☐ Real Estate
☐ Telecom/Wireless
☐ Transportation/Delivery
☐ Other (please specify) _____

UAS Platform

Q3.1: On which specific UAS platform(s) do you operate? (select all that apply)
Drone:

☐ DJI ☐ Parrot ☐ Yuneec ☐ Draganfly ☐ Lockheed Martin
☐ Micropilot ☐ ArduPilot ☐ Pixhawk
☐ Other (please specify) _____

Ground Control Station

☐ DJI GO ☐ Autopilot ☐ Litchi ☐ Mission Planner ☐ QGroundControl
☐ UgCS ☐ Lockheed Martin UGCS
☐ Other (please specify) _____

Operational Data Platform

☐ DJI FlightHub ☐ AirData ☐ DroneLogbook ☐ Kittyhawk ☐ Skyward
☐ Other (please specify) _____

Q3.2: What functions are you mostly using in a ground control station? (select all that apply)
☐ Drone and payloads database and specifications
☐ UAS configuration/tuning
☐ Flight plan and waypoints generation
☐ Flight monitoring and telemetry data logging
☐ Multiple drone mission
☐ Advanced mission planning: surveying, obstacle avoidance
☐ Drone simulation
☐ Flight plan simulation
☐ Other (please specify) _____

UAS Safety

Q4.1: How often do you experience an in-flight failure (of any kind) while operating your UAS?
☐ 0-10% of the time
☐ 11-25% of the time
☐ 26-50% of the time
☐ 51-75% of the time

☐ 76-90% of the time
☐ 91-100% of the time

Q4.2: What areas have been most relevant for improving safety during your UAS operations? (select all that apply)
☐ Understanding air navigation charts and airspace
☐ Developing and sustaining a maintenance program
☐ Use of Aeronautical Decision Making risk assessment tools
☐ Greater familiarity with weather effects on UAS operations
☐ Allocation of resources for flight planning
☐ Other (please specify) _____

Q4.3: In which area(s) do you find it most challenging to maintain compliance with FAA Part 107? (Select up to 3)
☐ Equipment logs and records
☐ Remote pilot flight logs
☐ Airspace access
☐ Weather conditions
☐ Flight in congested areas (directly over populace)
☐ VLOS (Visual Line of Sight)
☐ Other (please specify) _____

UAS Tools/Services

Q5.1: Are you using any of the UAS services listed below?
☐ Cloud-based data storage and access
☐ Flight Logbook
☐ Drone fleet management
☐ Drone repair/maintenance
☐ Equipment supply (platforms, sensors, components)
☐ Image processing
☐ Mission planning and scheduling
☐ Accident and incident reporting
☐ Risk assessment
☐ Flight simulation
☐ Other (please specify) _____

Q5.2: How important are the following areas to your UAS operations? (please use number 1-10 to rank them)
____ Drone asset management
____ Drone specification and supplier data
____ Drone maintenance history
____ Operation team management
____ Flight logbook
____ Single drone mission planning
____ Multi-drone mission planning and scheduling
____ Multi-drone control
____ Real time data processing (e.g. Image processing)
____ Post flight data processing
____ Accident and incident reporting
____ Risk assessment
____ Flight simulation
____ Other (please specify) _____

Q5.3: Please tell more about your expectations from the currently available tools and services for your UAS operations?

REFERENCES

[1] Adolphs, P., Bedenbender, H., Dirzus, D., Ehlich, M., Epple, U., Hankel, M., Heidel, R., Hoffmeister, M., Huhle, H., Kärcher, B., Koziolek, H., Pichler, R., Pollmeier, S., Schewe, F., Walter, A., Waser, B., and Wollschlaeger, M., 2015, "Status Report: Reference Architecture Model Industrie 4.0 (RAMI4.0)," VDI/VDE and ZVEI.

[2] Allmendinger, G., and Lombreglia, R., 2005, "Four Strategies for the Age of Smart Services," Harvard Business Review, Vol. 83, No. 10, pp. 131-145.

[3] Ameri, F., and Dutta, D., 2005, "Product Lifecycle Management: Closing the Knowledge Loops," Computer-Aided Design and Applications, Vol. 2, No. 5, pp. 577-590.

[4] Anandkumar, A., Ge, R., Hsu, D., and Kakade, S.M., 2014, "A Tensor Approach to Learning Mixed Membership Community Models," Journal of Machine Learning Research, Vol. 15, No. 1, pp. 2239-2312.

[5] Asmita, S., and Shukla, K.K., 2014, "Review on the Architecture, Algorithm and Fusion Strategies in Ensemble Learning," International Journal of Computer Applications, Vol. 108, No. 8, pp. 21-28.

[6] Assuncao, M.D., Calheiros, R.N., Bianchi, S., Netto, M.A.S., and Buyya, R., 2015, "Big Data computing and clouds: Trends and future directions," Journal of Parallel and Distributed Computing, Vol. 79-80, pp. 3-15.

[7] Aurich, J.C., Wolf, N., Siener, M., and Schweitzer, E., 2009, "Configuration of product-service systems," Journal of Manufacturing Technology Management, Vol. 20, No. 5, pp. 591-605.

[8] Austin, S., Baldwin, A., Li, B., and Waskett, P., 1999, "Analytical design planning technique: a model of the detailed building design process," Design Studies, Vol. 20, No. 3, pp. 279-296.

[9] Baines, T.S., Lightfoot, H.W., Evans, S., Neely, A., Greenough, R, Peppard, J., Roy, R., Shehab, E., Braganza, A., Tiwari, A., Alcock, J.R., Angus, J.P., Bastl, M., Cousens, A., Irving, P., Johnson, M., Kingston, J., Lockett, H., Martinez, V., Michele, P., Tranfield, D., Walton, I.M., and Wilson, H., 2007, "State-of-the-art in product-service systems," Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, Vol. 221, No. 10, pp. 1543-1552.

[10] Baldwin, C.Y., and Clark, K.B., 2006, "Modularity in the Design of Complex Engineering Systems," In: Braha, D., Minai A., Bar-Yam Y. (eds) Complex Engineered Systems, Understanding Complex Systems, Springer, Heidelberg, Berlin, Germany.

[11] Barbau, R., Krima, S., Rachuri, S., Narayanan, A., Fiorentini, X., Foufou, S., and Sriram, R.D., 2012, "OntoSTEP: Enriching product model data using ontologies," Computer-Aided Design & Applications, Vol. 44, No. 6, pp. 575-590.

[12] Bartolomei, J.E., 2007, "Qualitative Knowledge Construction for Engineering Systems: Extending the

Design Structure Matrix Methodology in Scope and Procedure," Massachusetts Institute of Technology, Engineering Systems Division. Doctoral Dissertation.

[13] Bartolomei, J.E., Cokus, M., Dahlgren, J., de Neufville, R., Maldonado, D., and Wilds, J., 2007, "Analysis and Applications of Design Structure Matrix, Domain Mapping Matrix, and Engineering System Matrix Frameworks," Engineering Systems Division, Massachusetts Institute of Technology, Cambridge, MA, USA.

[14] Barton, J.D., 2010, "Fundamentals of Small Unmanned Aircraft Flight," Johns Hopkins APL Technical Digest, Vol. 31, No. 2, pp. 132-149.

[15] Bask, A., Lipponen, M., Rajahonka, M., and Tinnila, M., 2009, "The concept of modularity: diffusion from manufacturing to service production," Journal of Manufacturing Technology Management, Vol. 21, No. 3, pp. 355-375.

[16] Batchelor, J., and Andersen, H.R., 2012, "Bridging the Product Configuration Gap between PLM and ERP - An Automotive Case Study," Proceedings of the 19th International Product Development Management Conference, Manchester, UK.

[17] Beetz, M., Buss, M., and Wollherr, D., 2007, "Cognitive Technical Systems – What Is the Role of Artificial Intelligence?" Proceedings of the 30th Annual German Conference on AI, Osnabrück, Germany, pp. 19-42.

[18] Berthold, M.R., Cebron, N., Dill, F., Gabriel, T.R., Koetter, T., Meinl, T., Ohl, P., Thiel, K., and Wiswedel, B., 2009, "KNIME: The Konstanz Information Miner, Version 2.0 and Beyond," ACM SIGKDD Explorations Newsletter, Vol. 11, No.1, pp. 26-31.

[19] Böhmann, T., Junginger, M., and Kremar, H., 2003, "Modular Service Architectures: A Concept and Method for Engineering IT Services," Proceedings of the 36th International Conference on System Sciences, Big Island, HI, USA.

[20] Briggs, C., Brown, G., Siebenaler, D, Faoro, J., and Rowe, S., 2010, "Model Based Definition," 51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Orlando, Florida, USA.

[21] Browning, T.R., 2001, "Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions," IEEE Transaction on Engineering Management, Vol. 48, No. 3, pp. 292-306.

[22] Castanedo, F., 2013, "A Review of Data Fusion Technique," The Scientific World Journal, Vol. 2013, pp. 1-19.

[23] Chandrasegaran, S.K., Ramani, K., Sriram, R.D., Horvan, I., Bernard, A., Harik, R.F., and Gao, W., 2013, "The evolution, challenges, and future of knowledge representation in product design systems," Computer-Aided Design, Vol. 45, No.2, pp. 204-228.

[24] Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., and Wirth, R., 2000, "CRISP-

DM 1.0: Step-by-step data mining guide," SPSS, CRISPMWP-1104.

[25] Chu, R., Duling, D., and Thompson, W., 2007, "Best Practices for Managing Predictive Models in a Production Environment," SAS Global Forum 2007.

[26] CIMdata, 2012a, "Analytics, PLM Converge Amid Data Tsunami; A Ceaseless Search for Sustainable Advantage," CIMdata, Inc.

[27] CIMdata., 2012b, "Product Lifecycle Management and the Data Deluge: Transforming Data to Enhance Performance," CIMdata, Inc.

[28] CIMdata., 2014, "Aras Innovator: Redefining Customization & Upgrades," CIMdata Commentary.

[29] Clements, P., and Northrop, L., 2001, "Software Product Lines: Practices and Patterns," 3rd Edition, Addison-Wesley Professional. ISBN: 978-0-201-70332-0.

[30] Colin, S., 2017, "The Current State of the Drone Industry," [online] https://www.slideshare.net/ColinSnow/current-state-of-the-drone-industry-skylogic-research (accessed November 14, 2018)

[31] Collins, S.T., Yassine, A.A., and Borgatti, S.P., 2008, "Evaluating Product Development Systems Using Network Analysis," Systems Engineering, Vol. 12, No. 1, pp. 55-68.

[32] Coltheart, M., 1999, "Modularity and cognition," Trends in Cognitive Sciences, Vol. 3, No. 3, pp. 115-120.

[33] Conforto, E.C., and Amaral, D.C., 2016, "Agile project management and stage-gate model - A hybrid framework for technology-based companies," Journal of Engineering and Technology Management, Vol. 40, pp. 1-14.

[34] Cooper, R.G., 1994, "Perspective: Third-Generation New Product Processes," Journal of Product Innovation Management, Vol. 11, No. 1, pp. 3-14.

[35] Cooper, R.G., 2008, "Perspective: The Stage-Gate Idea-to-Launch Process – Update, What's New, and NexGen Systems," Journal of Product Innovation Management, Vol. 25, No. 3, pp. 213-232.

[36] Cooper, R.G., 2014, "What's Next? After Stage-Gate," Research-Technology Management, Vol. 57, No. 1, pp. 20-31.

[37] Cooper, R.G., 2016, "Agile-Stage-Gate Hybrids," Research-Technology Management, Vol. 59, No. 1, pp. 21-29.

[38] Danilovic, M., and Browning, T.R., 2007, "Managing complex product development projects with design structure matrices and domain mapping matrices," International Journal of Project Management, Vol. 25, No. 3, pp. 300-314.

[39] Dasarathy, B.V., 1997, "Sensor Fusion Potential Exploitation – Innovative Architectures and Illustrative Applications," Proceedings of the IEEE, Vol. 85, No. 1, pp. 24–38.

[40] Deb, S., Parra-Castillo, J.R., and Ghosh, K., 2011, "An Integrated and Intelligent Computer-Aided Process Planning Methodology for Machined Rotationally Symmetrical Parts," International Journal of Advanced

Manufacturing Systems, Vol. 13, No. 1, pp. 1-26.

[41] Debevoise, T., and Taylor, J., 2014, “The MicroGuide to Process and Decision Modeling in BPMN/DMN: Buidling More Effective Processes by Integrating Process Modeling with Decision Modeling,” CreateSpace Independent Publishing Platform, USA. ISBN: 978-1-502-78964-8.

[42] Decker, R., and Stummer, C., 2017, “Marketing Management for Consumer Products in the Era of the Internet of Things,” Advances in Internet of Things, Vol. 2017, No. 7, pp. 47-70.

[43] Dhar, V., 2013, “Data Science and Prediction,” Communications of the ACM, Vol. 56, No. 12, pp. 64-73.

[44] Dietterich, T.G., 2000, “Ensemble Methods in Machine Learning,” Lecture Notes in Computer Science, Multiple Classifier Systems (MCS2000), vol. 1857, pp. 1-15.

[45] Distanont, A., Haapasalo, H., Kamolvej, T., and Meeampol, S., 2012, “Interaction Patterns in Collaborative Product Development (CPD),” International Journal of Synergy and Research, Vol. 1, No. 2, pp. 21-43.

[46] DKE., 2014, “The German Standardization Roadmap Industry 4.0,” DKE German Commission for Electrical, Electronic & Information Technologies of DIN and VDE.

[47] Durrant-Whyte, H.F., 1988, “Sensor models and multisensor integration,” International Journal of Robotics Research, Vol. 7, No. 6, pp.97–113.

[48] EIU: Economist Intelligent Unit, 2015, “Developing smart products,” Survey Report. [online] http://www.economistinsights.com/technology-innovation/analysis/developing-smart-products (Accessed November 14, 2018)

[49] Eklund, U., and Bosch, J., 2012, “Applying Agile Development in Mass-Produced Embedded Systems,” Proceedings of the International Conference on Agile Software Development (XP 2012), Malmö, Sweden, pp.31-46.

[50] Eklund, U., Holmström, O.H., and Strøm, N.J., 2014, “Industrial Challenges of Scaling Agile in Mass-Produced Embedded Systems,” Proceedings of the International Conference on Agile Software Development (XP 2014), Rome, Italy, pp. 30-42.

[51] Elbanhawi, M., and Simic, M., 2014, “Sampling-Based Robot Motion Planning: A Review,” IEEE Access, Vol. 2, pp. 56-77.

[52] Fenves. S.J., 2002, “A Core Product Model for Representing Design Information,” National Institute of Standards and Technology, Gaithersburg, MD, USA. (NISTIR6736).

[53] Fisher, A., Nolan, M., Friedenthal, S., Loeffler, M., Sampson, M., Bajaj, M., VanZandt, L., Hovey, K., Palmer, J., and Hart, L., 2014, “Model Lifecycle Management for MBSE,” INCOSE International Symposium, Vol. 24, No.1, pp. 207-229.

[54] Fricke, E., and Schulz, A.P., 2005, “Design for changeability (DfC): Principles to enable changes in systems throughout their entire lifecycle,” Systems Engineering, Vol. 8, No. 4, pp. 342-359.

[55] Gauvin, L., Panisson, A., and Cattuto, C., 2014, “Detecting the Community Structure and Activity Patterns

of Temporal Networks: A Non-Negative Tensor Factorization Approach," PLOS ONE, Vol. 9, No. 1, pp. 1-13.

[56] Geiger, C., and Sarakakis, G., 2016, "Data driven design for reliability," Proceedings of the 2016 Annual Reliability and Maintainability Symposium (RAMS), Tucson, AZ, USA.

[57] Geisberger, E., and Broy, M. (Eds.), 2014, "Living in a networked world," Integrated research agenda Cyber-Physical Systems (agendaCPS)(acatech STUDY), Herbert Utz Verlag, Munich, Germany.

[58] Gens, F., 2013, "The 3rd Platform: Enabling Digital Transformation," International Data Corporation (IDC).

[59] Gericke, K., and Blessing, L., 2011, "Comparisons of Design Methodologies and Process Models across Disciplines: A Literature Review," Proceedings of the 18th International Conference on Engineering Design (ICED 11), Lyngby/Copenhagen, Denmark, Vol. 1, pp. 393-404.

[60] Gershenson, J.K., Prasad, G.J., and Allamneni, S., 1999, "Modular Product Design: A Life-cycle View," Journal of Integrated Design and Process Science, Vol. 3, No. 4, pp. 13-26.

[61] Gerwin, D., and Barrowman, N. J., 2002, "An Evaluation of Research on Integrated Product Development," Management Science, Vol. 48, No. 7, pp. 938-953.

[62] Gibbert, M., and Ruigrok, W., 2010, "The 'What' and 'How' of Case Study Rigor: Three Strategies Based on Published Work," Organizational Research Methods, Vol. 13, No. 4, pp. 710-737.

[63] Gifford, C., delaHostria, E., Noller, D., Childress, L., and Boyd, A., 2006, "Related Manufacturing Integration Standards, A Survey," MESA International, ISA, GE Fanuc Automation, Rockwell Automation, and IBM Corporation.

[64] Goedkoop, M.J., van Halen, C.J.G., te Riele, H.R.M., and Rommens, P.J.M., 1999, "Product Service Systems, Ecological and Economic Basics," Report No. 1999/36.

[65] Grieves, M., 2005, "Product Lifecycle Management: Driving the Next Generation of Lean Thinking," McGraw-Hill, New York, NY, USA. ISBN: 978-0-071-45230-4.

[66] Gupta, S.G., Ghonge, M.M., and Jawandhiya, P.M., 2013, "Review of Unmanned Aircraft System (UAS)," International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), Vol. 2, No. 4, pp. 1646-1658.

[67] Gutierrez, C., Garbajosa, J., Diaz, J., and Yague, A., 2013, "Providing a Consensus Definition for the Term 'Smart Product'", Proceedings of the 20th Annual IEEE International Conference and Workshops on the Engineering of Computer Based Systems (ECBS), Scottsdale, AZ, USA.

[68] Hehenberger, P., Vogel-Heuser, B., Bradley, D., Eynard, B., Tomiyama, T., and Achiche, S., 2016, "Design, modeling, simulation and integration of cyber physical systems: Methods and applications," Computers in Industry, Vol. 82, pp. 273-289.

[69] Helmer, R., Yassine, A., and Meier, C., 2008, "Systematic module and interface definition using component

design structure matrix," Journal of Engineering Design, Vol. 21, No. 6, pp. 647-675.

[70] Herrmann, J.W., and Schmidt, L.C., 2002, "Viewing Product Development as a Decision Production System," Proceedings of the ASME 2002 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Montreal, Canada, Vol. 4, pp. 323-332.

[71] Hölttä-Otto, K., and de Weck, O.L., 2007, "Degree of Modularity in Engineering Systems and Products with Technical and Business Constraints," Concurrent Engineering: Research and Applications, Vol. 15, No. 2, pp. 113-126.

[72] Huang, H.M., 2008, "Autonomy Levels for Unmanned Systems (ALFUS) Framework Volume I: Terminology," Special Publication, National Institute of Standards and Technology, Gaithersburg, MD, USA. (1011-I-2.0)

[73] Hubaux, A., Jannach, D., Drescher, C., Murta, L., Mannisto, T., Czarnecki, K., Heymans, P., Nguyen, T., and Zanker, M., 2012, "Unifying Software and Product Configuration: A Research Roadmap," Proceedings of the ECAI 2012 Workshop on Configuration, Montpellier, France.

[74] ISO: International Organization for Standardization, 2014, "ISO 10303-242:2014: Industrial automation systems and integration -- Product data representation and exchange -- Part 242: Application protocol: Managed model-based 3D engineering," ISO, Geneva, Switzerland.

[75] Ishii, K., 1997, "Modularity: A Key Concept in Product Life-cycle Engineering," Frontiers of Engineering: Reports on Leading Edge Engineering, pp. 17-22.

[76] Jain, J., Ari, I., and Li, J., 2008, "Understanding the challenges faced during the management of data mining models," Proceedings of the 2nd ACM Symposium on Computer Human Interaction for Management of Information Technology, New York, NY, USA

[77] Jarratt, T.A.W., Eckert, C.M., Caldwell, N.H.M., and Clarkson, P.J., 2010, "Engineering change: an overview and perspective on the literature," Research in Engineering Design, Vol. 22, No. 2, pp. 103-124.

[78] Jawahir, I.S., Rouch, K.E., Dillon, O.W., Holloway, L. Jr., Hall, A., and Knuf, J., 2007, "Design for Sustainability (DFS): New Challenges in Developing and Implementing a Curriculum for Next Generation Design and Manufacturing Engineers," International Journal of Engineering Education, Vol. 23, No. 6, pp. 1053-1064.

[79] Jiao, J., Tseng, M.M., Duffy, V.G., and Lin, F., 1998, "Product family modeling for mass customization," Computers & Industrial Engineering, Vol. 35, No. 3-4, pp. 495-498.

[80] Kaiser, L., Gomez, A.N., Shazeer, N., Vaswani, A., Parmar, N., Jones, L., and Uszkoreit, J., 2017, "One Model To Learn Them All," arXiv: 1706.05137 [cs.LG].

[81] Kalligeros, K., 2006, "Platforms and Real Options in Large-Scale Engineering Systems," Engineering Systems Division, Massachusetts Institute of Technology. Doctoral Dissertation.

[82] Kärkkäinen, M., Holmström, J., Främling, K., and Artto, K., 2003, "Intelligent products – a step towards a

more effective project delivery chain," Computers in Industry, Vol. 50, No. 2, pp. 141-151.

[83] Karlström, D., and Runeson, P., 2005, "Combining Agile Methods with Stage-Gate Project Management," Journal of IEEE Software, Vol. 22, No. 3, pp. 43-49.

[84] Karlström, D., and Runeson, P., 2006, "Integrating agile software development into stage-gate managed product development," Empirical Software Engineering, Vol. 11, No. 2, pp. 203-225.

[85] Kassner, L., Gröger, C., Mitschang, B., and Westkämper, E., 2015, "Product Life Cycle Analytics – Next Generation Data Analytics on Structured and Unstructured Data," Proceedings of the 9th CIRP Conference on Intelligent Computation in Manufacturing Engineering (CIRP ICME'14), Naples, Italy, 33, pp. 35-40.

[86] Kendoul, F., 2012, "Survey of Advances in Guidance, Navigation, and Control of Unmanned Rotorcraft Systems," Journal of Field Robotics, Vol. 29, No. 2, pp. 315-378.

[87] Kim, D.B., Shin, S.J., Shao, G., and Brodsky, A., 2015, "A decision-guidance framework for sustainability performance analysis of manufacturing processes," International Journal of Advanced Manufacturing Technology, Vol. 78, No. 9-12, pp. 1455-1471.

[88] Kiritsis, D., 2011, "Closed-loop PLM for intelligent products in the era of the Internet of things," Computer-Aided Design, Vol. 43, No. 5, pp. 479-501.

[89] Ko, J., and Fox, D., 2008, "GP-BayesFilters: Bayesian Filtering Using Gaussian Process Prediction and Observation Models," Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France.

[90] Kolda, T.G., 2006, "Multilinear Operators for Higher-Order Decompositions," Tech. Report, Sandia National Laboratories, Albuquerque, NM, and Livermore, CA. (SAND2006-2081)

[91] Kolda, T.G., and Bader, B.W., 2009, "Tensor Decompositions and Applications," SIAM Review, Vol. 51, No. 3, pp. 455-500.

[92] Kovar, D., and Bollo, J., 2018, "Drone Forensics," Digital Forensics Magazine, Vol. 34, pp. 14-19.

[93] Krishnan, V., and Ulrich, K.T., 2011, "Product Development Decisions: A Review of the Literature," Management Science, Vol. 47, No. 1, pp. 1-21.

[94] Krovi, R., Chandra, A., and Rajagopalan, B., 2003, "Information Flow Parameters for Managing Organizational Processes," Communications of the ACM, Vol. 46, No. 2, pp. 77-82.

[95] Kurgan, L.A., and Musilek, P., 2006, "A survey of Knowledge Discovery and Data Mining process models," The Knowledge Engineering Review, Vol. 21, No. 1, pp. 1-24.

[96] Lacher, A.R., Maroney, D.R., and Zeitlin, A.D., 2007, "Unmanned Aircraft Collision Avoidance – Technology Assessment and Evaluation," Technical Paper, the MITRE Corporation, McLean, VA.

[97] Langlois, R.N., 2000, "Modularity in technology and organization," Journal of Economic Behavior & Organization, Vol. 49, pp. 19-37.

[98] LaValle, S.M., and Kuffner, J.J. Jr., 2001, "Randomized Kinodynamic Planning," International Journal of

Robotics Research, Vol. 20, No. 5, pp. 378-400.

[99] Li, Y., and Roy, U., 2014, "A STEP-based Approach toward Cooperative Product Design for Sustainability," Proceedings of the 34th Computers and Information in Engineering Conference (ASME IDETC/CIE 2014), Buffalo, New York, USA.

[100] Li, Y., Roy, U., Lee, Y.T., and Rachuri, S., 2015a, "Integrating Rule-based Systems and Data Analytics Tools Using Open Standard PMML," Proceedings of the 35th Computers and Information in Engineering Conference (ASME IDETC/CIE 2015), Boston, Massachusetts, USA.

[101] Li, Y., Roy, U., Shin, S.J., and Lee, Y.T., 2015b, "A 'Smart Component' Data Model In PLM," Proceedings of the 2015 IEEE International Conference on Big Data, Santa Clara, California, USA.

[102] Li, Y., Roy, U., and Saltz, J.S., 2017a, "Modular Design of Data-Driven Analytics Models in Smart-Product Development," Proceedings of the ASME 2017 International Mechanical Engineering Congress & Exposition, Tampa, Florida, USA.

[103] Li, Y., Zhang, H., Roy, U., and Lee, Y.T., 2017b, "A Data-Driven Approach for Improving Sustainability Assessment in Advanced Manufacturing," Proceedings of the 2017 IEEE International Conference on Big Data, Boston, Massachusetts, USA.

[104] Li, Y., and Roy, U., 2018, "A Multi-Objective Optimization Methodology towards Product Design for Sustainability," International Journal of Strategic Engineering Asset Management, Vol. 3, No. 2, pp. 154-176.

[105] Lohr, S., 2011, "Ex-Apple Leaders Push the Humble Thermostat into the Digital Age," The New York Times. [online] http://www.nytimes.com/2011/10/25/technology/at-nest-labs-ex-apple-leaders-remake-the-thermostat.html (accessed November 14, 2018).

[106] Lu, J., Sookoor, T., Srinivasan, V., Gao, G., and Holben, B., 2010, "The Smart Thermostat: Using Occupancy Sensors to Save Energy in Homes," Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, Zurich, Switzerland, pp. 211-224.

[107] Lubell, J., Chen, K., Horst, J., Frechette, S., and Huang, P., 2012, "Model Based Enterprise/Technical Data Package," Summit Report, National Institute of Standards and Technology, Gaithersburg, MD, USA. (Technical Note 1753)

[108] MacDougall, W., 2013, "Industrie 4.0—Smart Manufacturing for the Future," Germany Trade and Invest (GTAI).

[109] Marbán, O., Segovia, J., Menasalvas, E., and Fernández-Baizán, C., 2009, "Toward data mining engineering: A software engineering approach," Information Systems, Vol. 34, No. 1, pp. 87-107.

[110] McFarlane, D., Sarma, S., Chirn, J.L., Wong, C.Y., and Ashton, K., 2003, "Auto ID systems and intelligent manufacturing control," Engineering Applications of Artificial Intelligence, Vol. 16, No. 4, pp. 365-376.

[111] Metzler, T., and Shea, K., 2010, "Cognitive Products: Definition and Framework," Proceedings of the 11th

International Design Conference – Design 2010, Dubrovnik, Croatia, pp. 865-874.

[112] Meyer, G.G., Främlingb, K., and Holmströmc, J., 2008, "Intelligent Products: A survey," Computers in Industry, Vol. 60, No. 3, pp. 137-148.

[113] Mikkola, J.H., and Gassmann, O., 2003, "Managing Modularity of Product Architectures: Toward an Integrated Theory," IEEE Transactions on Engineering Management, Vol. 50, No. 2, pp. 204-218.

[114] Mikkola, J.H., 2007, "Management of Product Architecture Modularity for Mass Customization: Modeling and Theoretical Considerations," IEEE Transactions on Engineering Management, Vol. 54, No. 1, pp. 57-69.

[115] Miller, T.D., and Elgard, P., 1998, "Defining Modules, Modularity and Modularization," Proceedings of the 13th IPS Research Seminar, Fuglsø, Denmark.

[116] Mittal, S., and Frayman, F., 1989, "Towards a generic model of configuration tasks," Proceedings of the 11th International Joint Conference on Artificial Intelligence, San Francisco, CA, USA, Vol. 2, pp. 1395-1401.

[117] Morris, H.D., Ellis, S., Feblowith, J., Knickle, K., and Torchia, M., 2014, "A Software Platform for Operational Technology Innovation," White Paper, International Data Corporation (IDC). (#249120)

[118] Mühlhäuser, M., 2008, "Smart Products: An Introduction," Constructing Ambient Intelligence, Vol. 11, pp. 158-164.

[119] Nannapaneni, S., Narayanan, A., Ak, R., Lechevalier, D., Sexton, T., Mahadevan, S., and Lee, Y.T., 2018, "Predictive Model Markup Language (PMML) Representation of Bayesian Networks: An Application in Manufacturing," Smart and Sustainable Manufacturing Systems, Vol. 2, No. 1., pp. 87-113.

[120] NATO., 2012, "Standard Interfaces of UAV Control System (UCS) for NATO UAV Interoperability," 3rd Edition, NATO Standardization Agency. (STANAG 4586)

[121] Nest Labs, 2012, "Nest Learning Thermostat Efficiency Simulation: Update Using Data from First Three Months," White Paper, Nest, Palo Alto, CA. [online] http://downloads.nest.com/efficiency_simulation_white_paper.pdf (accessed November 14, 2018)

[122] Nest Labs, 2013, "The Effect of Schedules on HVAC Runtime for Nest Learning Thermostat Users," White Paper, Nest, Palo Alto, CA.

[123] Nest Labs, 2014, "Enhanced Auto-Schedule," White Paper, Nest, Palo Alto, CA. [online] https://nest.com/downloads/press/documents/enhanced-auto-schedule-white-paper.pdf (accessed November 14, 2018)

[124] Nest Labs, 2015, "Thermal Model and HVAC Control," White Paper, Nest, Palo Alto, CA. [online] http://downloads.nest.com/press/documents/thermal-model-hvac-white-paper.pdf (accessed November 14, 2018)

[125] Newman, M.E.J., and Girvan, M., 2004, "Finding and evaluating community structure in networks,"

Physical Review E, Vol. 69, No. 2, pp. 1-15.

[126] Ore, J.P., Elbaum, S., Burgin, A., and Detweiler, C., 2015, "Autonomous Aerial Water Sampling," Journal of Field Robotics, Vol. 32, No. 8, pp. 1095-1113.

[127] Papalexakis, E.E., Faloutsos, C., and Sidiropoulos, N.D., 2016, "Tensors for Data Mining and Data Fusion: Models, Applications, and Scalable Algorithms," ACM Transactions on Intelligent Systems and Technology (TIST), Vol. 8, No. 2, pp. 16-44.

[128] Park, J., Lechevalier, D., Ak, R., Ferguson, M., Law, K.H., Lee, Y.T., and Rachuri, S., 2017, "Gaussian Process Regression (GPR) Representation in Predictive Model Markup Language (PMML)," Smart and Sustainable Manufacturing Systems, Vol, 1, No. 1, pp. 121-141.

[129] Pasqual, M.C., and de Weck, O.L., 2012, "Multilayer network model for analysis and management of change propagation," Research in Engineering Design, Vol. 23, No. 4, pp. 305-328.

[130] Patil, D.J., 2012, "Data Jujitsu: The art of turning data into product," O'Reilly Media, Inc., CA. ISBN: 978-1-449-34115-2. [online] http://radar.oreilly.com/2012/07/data-jujitsu.html (accessed November 14, 2018)

[131] Pavliscak, P., 2015, "Data-Informed Product Design," 1st ed., O'Reilly Media, Inc., CA. ISBN: 978-1-491-93129-5. [online] https://www.oreilly.com/ideas/data-informed-product-design (accessed November 14, 2018)

[132] PCAST, 2014, "Report to the President: Accelerating U.S. Advanced Manufacturing," The President's Council of Advisors on Science and Technology. [online] https://obamawhitehouse.archives.gov/sites/default/files/microsites/ostp/PCAST/amp20_report_final.pdf (Accessed November 14, 2018)

[133] Peffer, T., Pritoni, M., Meier, A., Aragon, C., and Perry, D., 2011, "How people use thermostats in homes: A review," Building and Environment, Vol. 46, No. 12, pp. 2529-2541.

[134] Ponti, M.P. Jr., 2011, "Combining Classifiers: From the Creation of Ensembles to the Decision Fusion," Proceedings of the 24th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T), Alagoas, Brazil.

[135] Porter, M.E., and Heppelmann, J.E., 2014, "How Smart, Connected Products Are Transforming Competition," Harvard Business Review, Issue November 2014, pp. 64-88.

[136] Porter, M.E., and Heppelmann, J.E., 2015, "How Smart, Connected Products Are Transforming Companies," Harvard Business Review, Issue October 2015, pp. 96-114.

[137] Quéva, M., Männistö, T., Ricci, L., and Probst, C.W., 2011, "Modelling Configuration Knowledge in Heterogeneous Product Families," Proceedings of the Workshop on Configuration (ConfWS 2011), Barcelona, Spain, pp. 9-18.

[138] Quintana, V., Rivest, L., Pellerin, R., Venne, F., and Kheddouci, F., 2010, "Will Model-based Definition replace engineering drawings throughout the product lifecycle? A global perspective from aerospace

industry," Computers in Industry, Vol. 61, No. 5, pp. 497-508.

[139] Rabanser, S., Shchur, O., and Günnemann, S., 2017, "Introduction to Tensor Decompositions and their Applications in Machine Learning," arXiv:1711.10781 [stat.ML].

[140] Re, M., and Valentini, G., 2012, "Ensemble methods: A review," in Way, M.J. et al. (Eds.), Advances in Machine Learning and Data Mining for Astronomy, Chapman & Hall/CRC, pp.563-594.

[141] Reece, S., and Roberts, S., 2010, "An Introduction to Gaussian Processes for the Kalman Filter Expert," Proceedings of the 13th International Conference on Information Fusion, Edinburgh, UK.

[142] Rowell, A., and Ballou, M.C., 2014, "Business Strategy: Integrating Mechanical, Electrical/Electronic, and Software Development in an Era of Smart Products," IDC Manufacturing Insights. (#MI245545)

[143] Roy, U., Li, Y., and Zhu, B., 2014, "Building a Rigorous Foundation for Performance Assurance Assessment Techniques for Smart Manufacturing Systems," Proceedings of the 2014 IEEE International Conference on Big Data, Washington, DC, USA.

[144] Roy, U., Zhu, B., Li, Y., Zhang, H., and Yaman, O., 2014, "Mining Big Data in Manufacturing: Requirement Analysis, Tools and Techniques," Proceedings of ASME 2014 International Mechanical Engineering Congress and Exposition, Montreal, Canada, Vol. 11, pp. V011T14A047.

[145] Sabin, D., and Weigel, R., 1998, "Product Configuration Frameworks-A Survey," Intelligent Systems and their Applications, IEEE, Vol. 13, No. 4, pp. 42-49.

[146] Sabou, M., Kantorovitch, J., Nikolov, A., Tokmakoff, A., Zhou, X., and Motta, E., 2009, "Position Paper on Realizing Smart Products: Challenges for Semantic Web Technologies," Proceedings of CEUR Workshop, Vol. 522, pp. 135-147.

[147] Saltz, J.S., 2015, "The Need for New Processes, Methodologies and Tools to Support Big Data Teams and Improve Big Data Project Effectiveness," Proceedings of the 2015 IEEE International Conference on Big Data, Santa Clara, CA, pp. 2066-2071.

[148] Sapp, C.E., 2017, "Preparing and Architecting for Machine Learning," Gartner Technical Professional Advice, pp. 1-37. (G00317328)

[149] Sarigecili, M.I., Roy, U., and Rachuri, S., 2014, "Interpreting the semantics of GD&T specifications of a product for tolerance analysis," Computer-Aided Design, Vol. 47, pp. 72-84.

[150] Schenk, M., and Seelmann-Eggebert, R., 2003, "Enabling mass customization across the value chain," IEE Seminar on Mass Customization: Turning Customer Differences into Business Advantages, London, UK.

[151] Seshia, S.A., Hu, S., Li, W., and Zhu, Q., 2017, "Design Automation of Cyber-Physical Systems: Challenges, Advances, and Opportunities," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 36, No. 9, pp. 1421-1434.

[152] Shearer, C., 2000, "The CRISP-DM Model: The New Blueprint for Data Mining," Journal of Data Warehousing, Vol. 5, No. 4, pp. 13-22.

[153] Shin, S., Woo, J., Kim, D.B., Kumaraguru, S., and Rachuri, S., 2016, “Developing a virtual machining model to generate MTConnect machine-monitoring data from STEP-NC,” International Journal of Production Research, Vol. 54, No. 15, pp. 4487-4505.

[154] Simpson, T.W., Maier, J.R., and Mistree, F., 2001, “Product platform design: method and application,” Research in Engineering Design, Vol. 13, No. 1, pp. 2-22.

[155] Simpson, T.W., Siddique, Z., and Jiao, J., 2005, “Platform-Based Product Family Development,” in Simpson, T.W. et al. (Eds.), Product Platform and Product Family Design: Methods and Applications, Springer, New York, NY, USA, pp. 1-15.

[156] Smith, D.J., 2013, “Power-by-the-hour: the role of technology in reshaping business strategy at Rolls-Royce,” Technology Analysis & Strategic Management, Vol. 25, No. 8, pp. 987-1007.

[157] SMLC, 2011, “Implementing 21st Century Smart Manufacturing,” Workshop Summary Report, Smart Manufacturing Leadership Coalition (SMLC).

[158] Steenstrup, K., Sallam, R.L., Eriksen, L., and Jacobson, S.F., 2014, “Industrial Analytics Revolutionizes Big Data in the Digital Business,” Gartner Research.

[159] Soininen, T., Tiihonen, J., Männistö, T., and Sulonen, R., 1998, “Towards a general ontology of configuration,” Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Vol. 12, No. 4, pp. 357-372.

[160] Solomatine, D.P., and Ostfeld, A., 2008, “Data-driven modelling: some past experiences and new approaches,” Journal of Hydroinformatics, Vol. 10, No. 1, pp. 3-22.

[161] Sosa, M.E., Eppinger, S.D., and Rowles, C.M., 2003, “Identifying Modular and Integrative Systems and Their Impact on Design Team Interactions,” Transactions of the ASME, Vol. 125, pp. 240-252.

[162] Sosa, M. E., Eppinger, S.D., and Rowles, C.M., 2004, “The Misalignment of Product Architecture and Organizational Structure in Complex Product Development,” Management Science, Vol. 50, No. 12, pp. 1674-1689.

[163] Subbarayan, S., 2005, “Integrating CSP Decomposition Techniques and BDDs for Compiling Configuration Problems,” Proceedings of the 2nd International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Prague, Czech Republic, pp. 351-365.

[164] Sudarsan, R., Fenves, S.J., Sriram, R.D., and Wang F., 2005, “A product information modeling framework for product lifecycle management,” Computer-Aided Design, Vol. 37, No. 13, pp. 1399-1411.

[165] Suh, N.P., 1990, “The Principles of Design,” Oxford University Press.

[166] Sun, K., Li, Y., and Roy, U., 2017, “A PLM-based Data Analytics Approach for Improving Product Development Lead Time in an Engineer-To-Order Manufacturing Firm,” Mathematical Modeling of Engineering Problems, Vol. 4, No. 2, pp. 69-74.

[167] Taylor, J., Fish, A., Vanthienen, J., and Vincent, P., 2013, “Emerging standards in decision modeling,” In iBPMS Expo: Intelligent BPM Systems: Impact and Opportunity, pp. 133-146.

[168] Tukker, A., 2004, “Eight types of product–service system: eight ways to sustainability? Experiences from SusProNet,” Business Strategy and the Environment, Special Issue: Innovating for Sustainability, Vol. 13, No. 4, pp. 246-260.

[169] Ulrich, K., 1995, “The role of product architecture in the manufacturing firm,” Research Policy, Vol. 24, No. 3, pp. 419-440.

[170] Ulrich, K.T., and Pearson, S., 1998, “Assessing the Importance of Design Through Product Archaeology,” Management Science, Vol. 44, No. 3, pp. 352-369.

[171] Ulrich, K.T., and Eppinger, S.D., 2012, “Product Design and Development, 5th ed.,” McGraw-Hill, NY, USA. ISBN: 978-0-073-40477-6.

[172] Unger, D.W., and Eppinger, S.D., 2009, “Comparing product development processes and managing risk,” International Journal of Product Development, Vol. 8, No. 4, pp. 382-402.

[173] Vandermerwe, S., and Rada, J., 1988, “Servitization of business: Adding value by adding services,” European Management Journal, Vol. 6, No. 4, pp. 314-324.

[174] Ventä, O., 2007, “Intelligent products and systems,” Technical report, VTT.

[175] Voss, C.A., and Mikkola, J.H., 2009, “Service Architecture and Modularity,” Decision Sciences, Vol. 40, No. 3, pp. 541-569.

[176] Wheatcraft, L.S., Ryan, M.J., and Svensson, C., 2017, “Integrated Data as the Foundation of Systems Engineering,” Proceedings of the 27th Annual INCOSE International Symposium (IS 2017), Adelaide, Australia, Vol. 27, No. 1, pp. 1423-1437.

[177] Wynn, D.C., and Clarkson, P.J., 2018, “Process models in design and development,” Research in Engineering Design, Vol. 29, No. 2, pp. 161-202.

[178] Yassine, A., 2004, “An Introduction to Modeling and Analyzing Complex Product Development Processes Using the Design Structure Matrix (DSM) Method,” Quaderni Di Management (Italian Management Review), Vol. 51, No. 9, pp. 1-17.

[179] Yin, H., Cam, L.L., and Roy, U., 2017, “Formation control for multiple unmanned aerial vehicles in constrained space using modified artificial potential field,” Mathematical Modelling of Engineering Problems, Vol. 4, No. 2, pp. 100-105.

[180] Yoo, M.J., Grozel, C., and Kiritsis, D., 2016, “Closed-Loop Lifecycle Management of Service and Product in the Internet of Things: Semantic Framework for Knowledge Integration,” Sensor 2016, Vol. 16, No. 7, pp. 1053-1078.

[181] Zha, X.F., and Sriram, R.D., 2006, “Platform-based product design and development: A knowledge-intensive support approach,” Knowledge-Based Systems, Vol. 19, No. 7, pp. 524-543.

[182] Zhu, L., Wang, Z., and Li, Z., 2018a, “Representing Time-Dynamic Geospatial Objects on Virtual Globes Using CZML – Part I: Overview and Key Issues,” International Journal of Geo-Information, Vol. 7, No. 97, pp. 1-21.

[183] Zhu, L., Wang, Z., and Li, Z., 2018b, “Representing Time-Dynamic Geospatial Objects on Virtual Globes Using CZML – Part II: Impact, Comparison, and Future Developments,” International Journal of Geo-Information, Vol. 7, No. 102, pp. 1-18.

VITA - YUNPENG LI

263 Link Hall, Syracuse, NY 13244
+1 315-380-6167 | yli179@syr.edu

EDUCATION

SYRACUSE UNIVERSITY **Syracuse, NY**

Doctoral Degree in Mechanical Engineering *January 2014 – December 2018*

- Dissertation: "A Smart Products Lifecycle Management (sPLM) Framework – Modeling for Conceptualization, Interoperability, and Modularity."
- GPA: 4.0/4.0

SYRACUSE UNIVERSITY **Syracuse, NY**

Master's Degree in Mechanical Engineering *August 2012 – December 2013*

- Capstone Project: "A Multi-Objective Optimization Methodology towards Product Design for Sustainability."
- GPA: 4.0/4.0

BEIHANG UNIVERSITY **Beijing, China**

Bachelor's Degree in Flight Vehicle Propulsion Engineering *September 1996 – June 2000*

- Graduate Cum Laude

PROFESSIONAL EXPERIENCE

USPLM, INC. **Syracuse, NY**
Co-Founder *January 2018 – Present*

- Lead to develop a scalable, cloud-based drone fleet management software, integrating UAS (Unmanned Aircraft System), PLM (Product Lifecycle Management), and IoT (Internet-of-Things) technologies.

SYRACUSE UNIVERSITY **Syracuse, NY**
Research Assistant *January 2014 – December 2017*

- Research in data models, process models, interoperability standards, and modular architecture to accommodate real-time data analytics in smart products and manufacturing systems. A *Smart Products Lifecycle Management (sPLM)* framework was developed to support integrated product development involving physical components and data-analytics models. (Sponsored by the U.S. National Institute of Standards and Technology).
- Design and development of a PLM architecture for UAS. The UAS-PLM is based on the sPLM concept and integrates mission planning tools and data analytics tools with CAD tools and other modeling and simulation tools. (Sponsored by the New York State government and the CASE center at Syracuse University).
- Collaborated with a local industrial liquid filtration system company in applying the sPLM concept to improve its product development lead time, using model-based optimization and advanced data visualization. (Sponsored by the CASE center at Syracuse University).

- Research in data models, data exchange standards, and sustainability assessment methods involved in sustainable manufacturing. (Sponsored by the U.S. National Institute of Standards and Technology).

DASSAULT SYSTÈMES SOLIDWORK CORPORATION **Beijing/Shanghai, China**
PDM Product Manager, Greater China *January 2008 – July 2012*

- Business development of the *SolidWorks Enterprise PDM* product in Greater China market.
- Collaborated with colleagues in marketing and sales, technical, and after-sales service departments, as well as solution partners, to support channel resellers and strategic customers in the region.
- Recruit, train, and certify technical teams for SolidWorks Enterprise PDM implementation in the region.

Territory Technical Manager, Western China *May 2007 – December 2007*

- Technical development, training, support of SolidWorks CAD and simulation products for channel resellers in the Western China territory.

DIGITAL CHINA INNOVATIVE SOLUTION LTD. **Beijing, China**
PDM Consultant *March 2007 – April 2007*

- Worked as a consulting team member focused on product data management (PDM) solution and CAD/CAM integration.

BEIJING AEROSPACE COMPUTER SOFTWARE CO., LTD **Beijing, China**
CAD/CAM Consultant *July 2000 – July 2006*

- Led a technical team to deliver CAD/CAM solution architecture, training, customization, and integration with enterprise systems, based on a portfolio of CAD/CAM software (Cimatron, CATIA V5, and TopSolid).

AFFLIATIONS AND AWARDS

- $250,000 Funding Award from the Genius NY Program (the world's largest business competition for unmanned systems), *2018*
- Active members of ASME, IEEE, INCOSE, PDMA, and AUVSI
- Membership nominations of Phi-Beta-Delta, Golden Key, and Phi-Kappa-Phi Honor Societies, *2013*
- "100% Club" Awards from Dassault Systèmes SolidWorks Corporation, *2010 and 2011*
- "Star Performance" Awards from Dassault Systèmes SolidWorks Corporation, *2008 and 2009*
- "President Club" Award from Dassault Systèmes SolidWorks Corporation, *2008*

CERTIFICATIONS AND PROFESSIONAL SKILLS

- PMP: Project Management Professional, from Project Management Institute, *2018*
- NPDP: New Product Development Professional, from Product Development and Management Association, *2017*
- UAS Remote Pilot Certificate, from Federal Aviation Administration, *2017*
- CPPA: Certified PDM Professional Administrator, from Dassault Systèmes SolidWorks Corporation, *2008*
- CSWP: Certified SolidWorks Professional, from Dassault Systèmes SolidWorks Corporation, *2007*

- CAD/CAM: SolidWorks, CATIA V5, TopSolid, Cimatron, SolidCAM
- Modeling and Simulation: Matlab, Arena, AnyLogic
- Data Analytics: R, KNIME, RapidMiner

- Programming: C#, VB.NET, JavaScript, HTML
- Standards: ISO 10303 (STEP), MESA B2MML, DMG PMML, OMG BPMN/DMN
- Languages: English, Mandarin

PUBLICATIONS

- **Li, Y.**, Roy, U., and Saltz, J. S., 2018, "Towards an Integrated Process Model for New Product Development with Data-Driven Features (NPD^3)," *Research in Engineering Design*. (Under Review)
- **Li, Y.**, Roy, U., Shin, S. J., and Lee, Y. T., 2018, "Towards Development of a Smart Product Data Model for Next Generation of PLM Systems," *Smart and Sustainable Manufacturing Systems*. (Under Review)
- **Li, Y.**, and Roy, U., 2018, "A Multi-Objective Optimization Methodology towards Product Design for Sustainability," *International Journal of Strategic Engineering Asset Management*, 3(2), pp. 154-176.
- **Li, Y.**, Zhang, H., Roy, U., and Lee, Y. T., 2017, "A Data-Driven Approach for Improving Sustainability Assessment in Advanced Manufacturing," *Proceedings of the 2017 IEEE International Conference on Big Data*, Boston, Massachusetts, USA.
- **Li, Y.**, Roy, U., and Saltz, J. S., 2017, "Modular Design of Data-Driven Analytics Models in Smart-Product Development," *Proceedings of the ASME 2017 International Mechanical Engineering Congress & Exposition*, Tampa, Florida, USA.
- **Li, Y.**, Roy, U., and Saltz, J. S., 2017, "An Initial View of a Process Model for Data-Driven Product Concept Development," *C+CA: Progress in Engineering and Science*, 42(3), pp. 982-987.
- **Li, Y.**, Roy, U., Shin, S. J., and Lee, Y. T., 2015, "A 'Smart Component' Data Model in PLM," *Proceedings of the 2015 IEEE International Conference on Big Data*, Santa Clara, California, USA.
- **Li, Y.**, Roy, U., Lee, Y. T., and Rachuri, S., 2015, "Integrating Rule-based Systems and Data Analytics Tools Using Open Standard PMML," *Proceedings of the ASME 35th Computers and Information in Engineering Conference*, Boston, Massachusetts, USA.
- **Li, Y.**, and Roy, U., 2015, "Challenges in Developing a Computational Platform to Integrate Data Analytics with Simulation-based Optimization," *Proceedings of the ASME 35th Computers and Information in Engineering Conference*, Boston, Massachusetts, USA.
- **Li, Y.**, and Roy, U., 2014, "A STEP-based Approach toward Cooperative Product Design for Sustainability," *Proceedings of the ASME 34th Computers and Information in Engineering Conference*, Buffalo, New York, USA.
- **Li, Y.**, and Roy, U., 2013, "Function-to-form Mapping in Design Synthesis Taking Sustainable Manufacturing Issues into Consideration," *Proceedings of the ASME 33th Computers and Information in Engineering Conference*, Portland, Oregon, USA.
- Sun, K., **Li, Y.**, and Roy, U., 2017, "A PLM-based Data Analytics Approach for Improving Product Development Lead Time in an Engineer-To-Order Manufacturing Firm," *Mathematical Modeling of Engineering Problems*, 4(2), pp. 69-74.
- Zhang, H., Zhu, B., **Li, Y.**, Yaman, O., and Roy, U., 2015, "Development and Utilization of a Process-oriented Information Model for Sustainable Manufacturing," *Journal of Manufacturing Systems*, 37(2), pp. 459-466.
- Roy, U., and **Li, Y.**, 2014, "Sustainability Assessment of the Injection Molding Process and the Effects of Material Selection," *Proceedings of the ASME 19th Design for Manufacturing and the Life Cycle Conference*, Buffalo, New York, USA.
- Roy, U., **Li, Y.**, and Zhu, B., 2014, "Building a Rigorous Foundation for Performance Assurance Assessment Techniques for Smart Manufacturing Systems," *Proceedings of the 2014 IEEE International Conference on Big Data*, Washington, DC, USA.
- Roy, U., Zhu, B., **Li, Y.**, Zhang, H., and Yaman, O., 2014, "Mining Big Data in Manufacturing: Requirement Analysis, Tools and Techniques," *Proceedings of the ASME 2014 International Mechanical Engineering Congress & Exposition*, Montreal, Canada.

REVIEW WORKS

- [Book Proposal] Kotu, V. (Yahoo), and Deshpande, B. R. (IBM), "Data Science: Concepts and Practice with RapidMiner," ELSEVIER. (Reviewed on April 2017)
- [Journal] Zhang, H., Calvo-Amodio, J., and Haapala, K. R., 2015, "Establishing foundational concepts for sustainable manufacturing systems assessment through systems thinking," *International Journal of Strategic Engineering Asset Management*, 2(3), pp. 249-269. [Conference] Perez, C. A. G., and Medellin-Castillo, H. I., 2017, "Computer Assisted

Design and Structural Topology Optimization of a Customized Craniofacial Implant," *Proceedings of the ASME 2017 International Mechanical Engineering Congress and Exposition*, Tampa, Florida, USA.

- [Conference] Fujii, S., and Aoyama, H., 2017, "Method for Producing Three-Dimensional Designs in Film Insert Molding," *Proceedings of the ASME 2017 International Mechanical Engineering Congress and Exposition*, Tampa, Florida, USA.
- [Conference] Ren, H, and Mo., W., 2014, "Breadth First Search Based Cosine Software Code Framework Automation Algorithm," *Proceedings of the ASME 34th Computers and Information in Engineering Conference*, Buffalo, New York, USA.
- [Conference] Nagiligari, B. K., Shah, J., Sha, Z., Thirugnanam, S., Jain, A., and Panchal, J., 2014, "Integrated Part Classification for Product Cost and Complexity Reduction," *Proceedings of the ASME 34th Computers and Information in Engineering Conference*, Buffalo, New York, USA.
- [Conference] Goepp, V., Rose, B., and Caillaud, E., 2013, "Performance Evaluation Model for Efficient Eco-Design Tool Integration," *Proceedings of the ASME 33rd Computers and Information in Engineering Conference*, Portland, Oregon, USA.

PATENTS

- **Li, Y.**, and Roy, U., 2018, "Smart Products Lifecycle Management Platform for Unmanned Aircraft Systems," *U.S. Provisional Patent Application No. 62/61385*, filed January 5, 2018.